THESIS

PLATFORM CAMERA AIRCRAFT DETECTION FOR APPROACH EVALUATION AND TRAINING

by

Ryan S. Yusko

March 2007

Thesis Adviser:	Mathias Kölsch
Co-Adviser:	Joe Sullivan

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE Platform Camera Aircraft Detection for Approach Evaluation and Training	**5. FUNDING NUMBERS** **R-9557**
6. AUTHOR(S) Ryan S. Yusko	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space and Naval Warfare Systems Center 49275 Electron Drive San Diego, CA 92152-5435	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Approach training currently relies solely on manual observation and verbal feedback to the pilot. This project aims to provide pilots and landing signal officers (LSOs) with valuable information about individual approaches in the carrier landing environment. The author investigated fully automatic flight path acquisition by means of computer vision-based analysis of platform camera video. The obtained data supports enhanced LSO training, real-time approach analysis, and pilot self-improvement through advanced review capabilities.

14. SUBJECT TERMS Computer Vision, Training, Approach Training, Pilot Training, LSO Training	**15. NUMBER OF PAGES** 83
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

PLATFORM CAMERA AIRCRAFT DETECTION FOR APPROACH EVALUATION AND TRAINING

Ryan S. Yusko
Lieutenant, United States Navy
B.S., United States Naval Academy, 1997

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN MODELING,
VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)**

from the

**NAVAL POSTGRADUATE SCHOOL
March 2007**

Author:	Ryan S. Yusko
Approved by:	Mathias Kölsch Thesis Advisor
	Joseph Sullivan Co-Advisor
	Rudolph Darken Chair, MOVES Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Approach training currently relies solely on manual observation of and verbal feedback to the pilot. This project aims to provide both pilots and landing signal officers (LSOs) with valuable information about individual approaches in the carrier landing environment. The author investigated fully automatic flight path acquisition by means of computer vision-based analyses of platform camera video. The obtained data supports enhanced LSO training, real-time approach analysis, and pilot self-improvement through advanced review capabilities.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

AAM	Active Appearance Model
BMP	Windows Bitmap Image Format
FCLP	Field Carrier Landing Practice
fps	frames per second
IFLOLS	Improved Fresnel Lens Optical Landing System
ISO	International Organization for Standardization
LSO	Landing Signal Officer
MPEG	Moving Picture Experts Group
MRY	Monterey Peninsula Airport
OpenCV	Open Source Computer Vision Library
RANSAC	Random Sample Consensus
ROI	Region of Interest
S-VHS	Super Video Home System
VHS	Video Home System
X3D	Extensible 3D
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language Transformation

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

Foremost, I wish to acknowledge the sacrifice and expert style advice provided by my mother. Without her input, this thesis read less clearly.

I wish to acknowledge the expertise and advice of my adviser, Dr. Mathias Kölsch. When I reported for duty at the Naval Postgraduate School, I lacked computer science background and computer vision experience: two of Dr. Kölsch's main areas of expertise. Dr. Kölsch's patience and perseverance allowed me to advance this research further than I had envisioned originally; always, I will welcome his advice.

Also, I wish to acknowledge the financial support of the Space and Naval Warfare Systems Center (SPAWAR), San Diego. Their funding provided premiere equipment for field testing, and covered travel expenses for this research. I could not have obtained equivalent results without such support.

Finally, I would like to acknowledge V-2 Division on board USS JOHN C. STENNIS (CVN-74)—specifically, IC2 Torres, who sacrificed hours from his busy work schedule to transfer 1:1 copies of S-VHS platform camera footage to VHS tape. Without his help, I could not have begun my research. Thank you.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Title: Platform Camera Aircraft Detection for Landing Approach Evaluation and Training

Abstract: Approach training currently relies solely on manual observation and verbal feedback to the pilot. This project aims to provide both pilots and landing signal officers (LSOs) with valuable information about individual approaches in the carrier landing environment. The project investigated fully automatic flight path acquisition by means of computer vision-based analysis of platform camera video. The obtained data supports enhanced LSO training, real-time approach analysis, and pilot self-improvement through advanced review capabilities.

Objectives: Provide additional feedback to all carrier pilots. Investigate computer vision methods to automatically extract an approach profile that can be saved in Extensible Markup Language (XML) for future review. Explore an ubiquitously accessible profile visualization based on this format. This approach visualization, combined with LSO comments/debriefing, has the potential to greatly enhance the aviator's carrier landing performance.

Determine feasibility of an airfield approach analysis. Reliable, automatically extracted landing profiles provide an objective frame of reference for LSO interpretation of airfield landings. Since no platform cameras are available on airfields, investigate computer vision-based detection and tracking of aircraft using stationary, on-field cameras. Consistent and easily available approach data provides positive reinforcement for pilots preparing to participate in ship-based evolutions.

Benefits: Consistent representations improve approach correlation among pilots and LSOs at the field and at the ship. This could cut training time and operating costs, improve approach quality and consistency, and ultimately increase boarding rate.

Archived approach data supports LSO training by providing examples and means for trend analysis. This data indicates to prospective LSOs typical approach problems.

Pilots can improve their own techniques using this supplemental approach feedback in conjunction with verbal LSO reports. Pilots could review their approaches at a later time to improve retention of personal mechanics.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. AREA OF RESEARCH

To supplement the Naval aviator’s approach feedback, this research aims to provide computer vision-based analyses of platform camera video for pilots and landing signal officers (LSOs). Valuable information captured and stored for further review aids in training, approach analysis, pilot self-improvement, or any of these in combination.

Research focused on three main questions:

1. Can a computer successfully detect and track, with two or three degrees of freedom, a moving aircraft in platform camera video?
2. Can the collected data produce usable representations of aircraft approaches?
3. Is it possible to extend a successful tracking implementation to a static airfield solution, providing LSOs consistent glideslope and centerline data during field carrier landing practice (FCLP)?

B. BACKGROUND

The U.S. Navy spends millions of dollars training pilots to land assets on aircraft carriers. Student pilots spend hundreds of hours preparing for first flights to the ship, and seasoned pilots practice hundreds of approaches to an airfield before each shipboard deployment/exercise. A proportional amount of money remains absent when providing pilots with consistent and reliable feedback, which they can use to improve their landing techniques.

Throughout the Naval aviator's career, the only feedback he receives about flown approaches comes from the LSO, an aviator trained in “signalling” aircraft progress (high, low, left right, etc.) during a pass. At the field during FCLPs, the LSO’s report of approach data can differ from the pilot’s observation of his pass. Rarely are LSO comments questioned due to the lack of objective information about the pilot’s approach. At the ship, a platform camera acts as a steady frame of reference for LSOs. One horizontal line and one vertical line projected into the camera's video denote glideslope and lineup, respectively. One need not be a pilot to realize how far an aircraft is off either parameter in its approach. Despite available visual aids, LSO teams often debate in closed sessions each aircraft’s performance. Remembering every pass accurately is challenging, due to lengthy recovery periods; at peak efficiency, one aircraft lands every 60 seconds. It is understandable that, with so many aircraft landing, and without receiving any breaks during landing periods, LSOs have a hard time capturing every approach in high detail. Despite visual aids present, the LSO's debrief of an approach to the carrier can fail to correlate with the pilot’s observations. In the ever-changing dynamic of landing at the ship, the pilot latches onto any information he can to improve his performance and crew safety. At the field and during shipboard operations, the LSO and the pilot’s observations might conflict. Unfortunately, a lack of correlation amounts to negative feedback, leaving junior pilots confused about how to improve approach techniques.

C. SCOPE

The main goal of this thesis is to provide additional feedback to all carrier pilots. If computer vision successfully detects, recognizes, and tracks the aircraft during its approach with at least two degrees of freedom, the profile of an aircraft's pass, saved for later review, is available to the pilot after receiving the LSO's comments and evaluation. This profile, combined with LSO debriefing, has the potential to greatly enhance the aviator's future carrier landing performance.

Potentially, LSOs benefit immensely from the airfield implementation for FCLPs. Computer vision has the ability to provide a standard of reference for LSO interpretation of an approach to any airfield. Unchanging camera references for both glideslope and lineup guarantee more accurate approach representations. Consistent depictions of aircraft approaches have the ability to provide positive reinforcement for pilots preparing to land at the ship.

D. ADVANTAGES

Providing supplemental objective feedback improves the pilot’s landing performance at the ship. Direct advantages include an increase in boarding rate, decreased mishap rates, and an increase in aircrew safety.

Approach information, stored in a standardized, extensible format, captures the pilot’s entire approach history, which is stored for future review and trend analysis.

Applied at the training level, results from the study have the potential to decrease aviator training time, increase carrier qualification rates, and save the Navy hundreds of thousands of dollars in fuel alone.

E. LIMITATIONS

Computer vision presents several limitations. Analysis of camera video must occur during daylight hours. Additionally, environmental visibility must be clear enough for the computer to successfully detect, recognize, and track the aircraft. This study produced high accuracy for both glideslope and lineup parameters; however, accuracy of range measurement (strictly through computer vision) proved too erratic for accurate approach representation.

THIS PAGE INTENTIONALLY LEFT BLANK

II. COMPUTER VISION BACKGROUND

A. INTRODUCTION

Significant progress continues through open-source development of computer vision-specific libraries. Originally, Intel developed a computer vision library, now available to researchers and commercial software developers as the Open Source Computer Vision Library (OpenCV). OpenCV focuses on real time computer vision, with specialized applications in such areas as object identification, segmentation and recognition, face recognition, motion tracking, and mobile robotics.

Unfortunately, an all-encompassing general-purpose algorithm does not exist for detecting object motion in video. There are several methods that, when used in tandem, can locate moving aircraft in video.

B. BASIC METHODS

1. Convolution

Convolution refers to applying a linear filter, through the use of a mask, across an image. By definition, convolving an image *f* of size MxN with a filter mask *w* of size *m*x*n* is given by:

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)\,f(x+s,y+t)$$

where:

$$a=\frac{m-1}{2}$$
$$b=\frac{n-1}{2}$$

To completely convolve the image, this equation must be applied for x = 0,1,2,…,M-1 and y = 0,1,2,…,N-1 (Gonzalez, 2002).

Often, g(x,y) is written as the response, *R*, produced from the *m*x*n* mask at any point (x,y) within an image. The formula becomes:

$$R = \sum_{i=1}^{mn} w_i z_i$$

where w_i represent mask coefficients and z_i are image values corresponding to those coefficients.

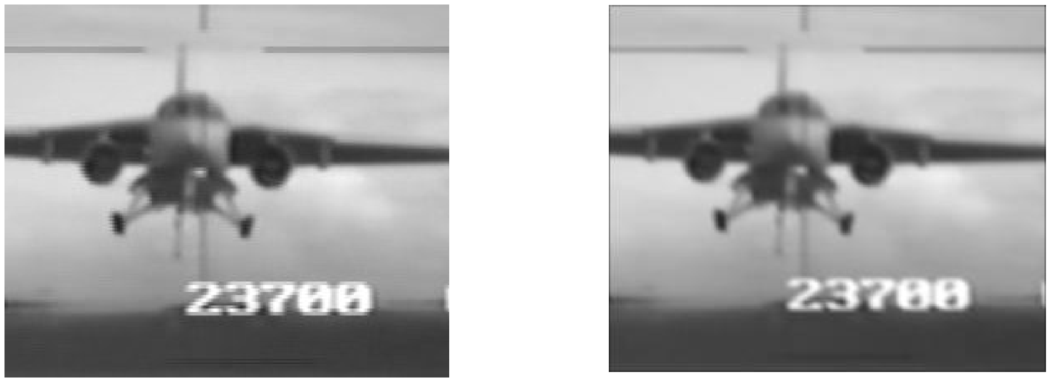


Figure 1. Simple convolution: Original image at left

Figure 1 shows the results of a simple convolution with the following kernel:

$$\begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 1/3 \end{bmatrix}$$

Convolution by itself does not provide useful aircraft detection; however, a specific kernel proves useful in preprocessing a noisy image with sharp or jagged edges.

2. Hough Transform

Typically, the Hough transform is used in computer vision applications to find lines or arbitrary shapes within an image (Ballard, 1981).

The Hough transform is a clustering technique that determines all parameters of the specified curve that passes through a pixel in the image. By iterating through each pixel, and tracking parameter values, which portions mostly satisfy the parameterized line equation are identified:

$$x \cdot \cos\theta + y \cdot \sin\theta + r = 0$$

Often called the line space, this set of (θ,r) pairs reveals the presence of a line, or of lines by identifying peaks in the accumulated parameter values. Figure 2 shows a sample Hough transform output that suggests presence of a line.

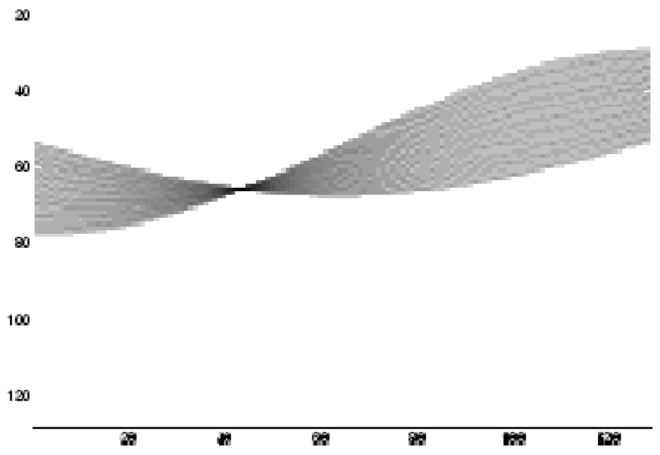


Figure 2. Sample Hough transform output

To cut processing time, the Hough transform sampling grid is varied; however, specifying an inappropriate grid size leads to quantization errors (Forsyth, 2003).

Hough transforms are sensitive to image noise; therefore, additional processing is necessary.

Initially, this research uses Hough transforms to locate the horizontal and vertical lines projected into the platform camera frame. The extra processing required, however, is unnecessary when locating and tracking the aircraft through successive frames of video.

3. Canny

The Canny algorithm (Canny, 1986) detects edges in an image.

Canny's algorithm focuses on intensity gradients within an image. The idea is simple: Higher gradients are more likely edges. To accomplish this, Canny first convolves an image with a Gaussian mask to eliminate noise. Then, through the use of several masks to detect horizontal, vertical, and diagonal edges, Canny forms the overall intensity gradient at every pixel in an image.

Finally, Canny traces the higher gradients to find edges. A universal threshold for all gradients does not exist to define edges. Canny utilizes hysteresis through two user-specified thresholds: high and low. The algorithm produces a binary image with defined pixels that represent edges within the original image.

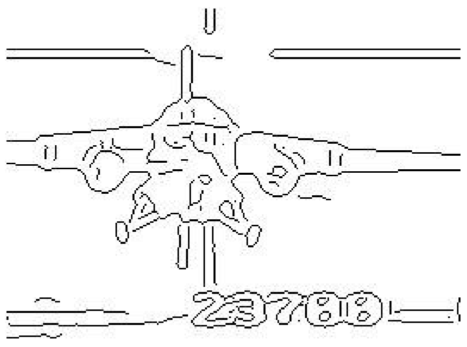


Figure 3. Canny algorithm applied to image from Figure 1

While the Canny algorithm produces images with object appearances easily identified by the human eye, they are unusable in training a computer to locate and track an aircraft in platform camera video.

4. Background Differencing

Background differencing, as in (Intille *et al.*, 1997) and (Fathy and Siyal, 1995), is a well-documented approach to locating object motion in successive frames of video. This method presents the first logical point for locating an aircraft in platform camera video.

When individual frames are subtracted from a learned background model, motion is seen:

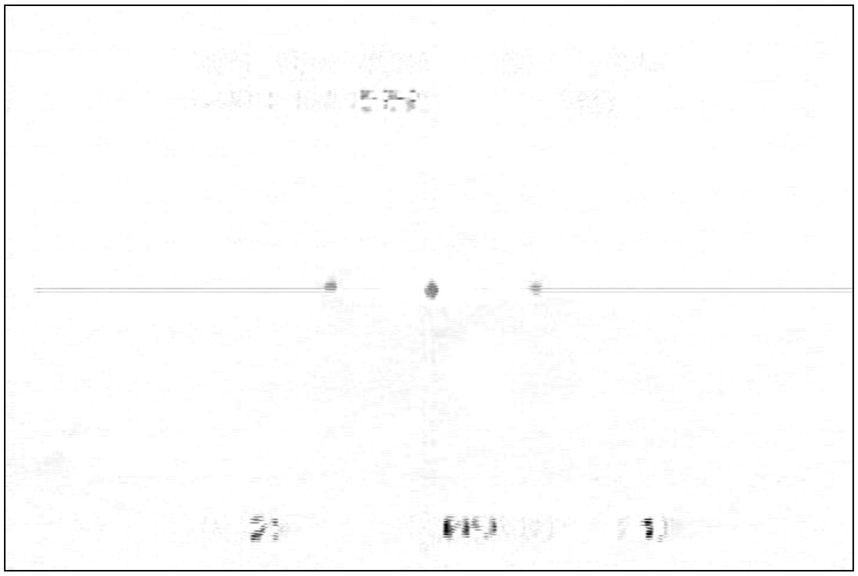


Figure 4. Background difference from a single frame of video (contrast enhanced)

In this work, difference images are thresholded, and simple morphological operations applied to enhance the information within them. This allows estimation of object locations with minimal overhead.

5. Morphology

Morphology provides critical tuning operations that refine noisy or otherwise problematic binary images. The four most basic morphology operations are dilation, erosion, opening, and closing.

Normally, morphology is performed on a binary image **B** by a structuring element **S**. Structurally, **S** usually has square (*m*x*m*) dimensions; the shape definition within can take any form. **S**'s dimensions are typically very small compared to **B**'s dimensions.

a. Dilation

Shapiro and Stockman (2001) define dilation as:

$$B \oplus S = \bigcup_{b \in B} S_b$$

While sweeping the structuring element **S** over the binary image **B**, the origin of **S** is compared to each individual pixel in **B**. If the origin of **S** matches a pixel in **B**, the contents of **S** are copied to the resultant image.

Dilation is used for filling gaps in images; it gradually adds more one-pixels to the output image when sweeping **B** with **S**.

b. Erosion

Shapiro and Stockman (2001) define erosion as:

$$B \ominus S = \left\{ b \middle| b + s \in B \forall s \in S \right\}$$

The erosion operation is similar to dilation, except as the structuring element is swept over the binary image, the output image only saves the one-pixel origin of **S** where all elements of **B** match **S**.

Erosion is most commonly used to eliminate unwanted noise in a processed frame or difference image.

c. Opening

Opening is defined as an erosion, followed by a dilation:

$$B \circ S = \left(B \ominus S \right) \oplus S$$

Openings smooth contours within difference images or eliminate small protrusions within the same.

d. Closing

Closing is defined as a dilation, followed by an erosion:

$$B \bullet S = \left(B \oplus S \right) \ominus S$$

Closings fill in holes and narrow valleys along edges in difference images.

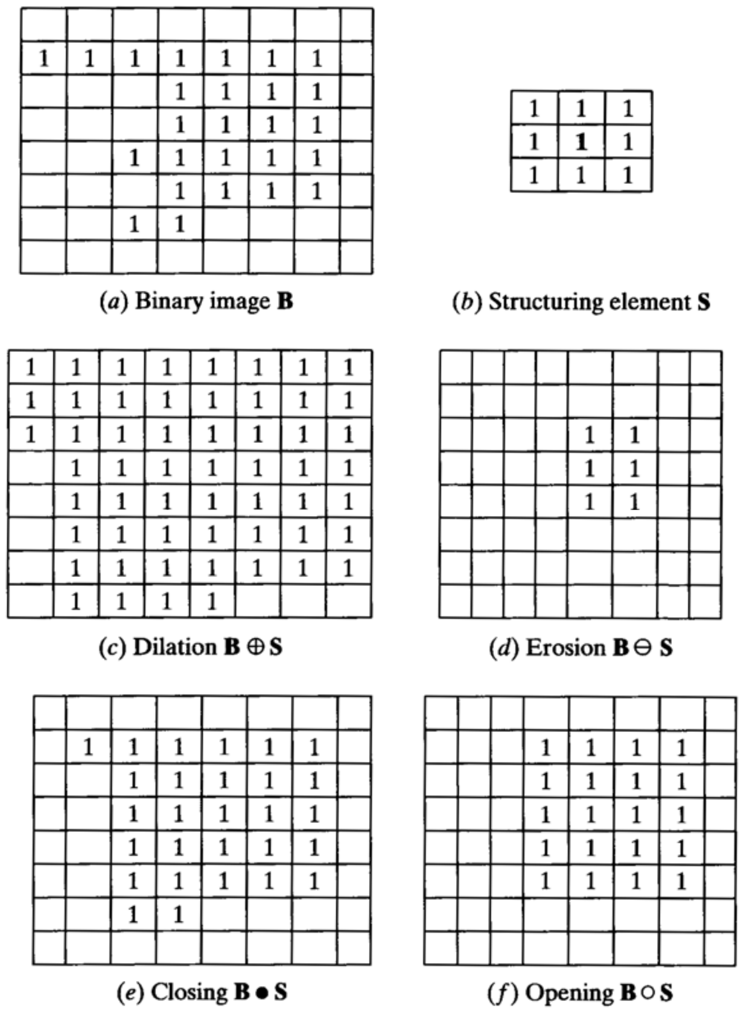


Figure 5. Simple morphology operations on a binary image **B** with structuring element **S** (from Shapiro and Stockman, 2001)

C. APPEARANCE-BASED METHODS

1. Boosted Cascade of Simple Features

Viola and Jones (2001) provide a rapid object detection method for frontal faces that runs at 15 frames per second on a conventional 700 MHz Intel Pentium III computer.

Their research introduced three major breakthroughs in the realm of computer vision-based object detection: an image representation called an *integral image*, a method for constructing a classifier by selecting a small number of features using AdaBoost, and a method for combining increasingly more complex classifiers in a “cascade” (Viola and Jones, 2001).

This research attempted to develop a highly effective machine-learned detection classifier for naval aircraft in platform camera video. While a Viola-Jones classifier showed promise in detecting aircraft, its implementation was beyond the scope of this thesis. A Viola-Jones-based method is recommended for future work.

2. Active Appearance Models

Cootes *et al.* (1998) described a method to match an image to a model in fewer iterations than typical near the turn of the century. Their method is primarily used for fitting deformable objects, but it is also useful in locating and/or identifying objects in an image.

Active Appearance Models (AAMs) rely on statistical models of the object's shape and grayscale appearance. Offline training establishes the relationship between a training image and a synthesized model example (Cootes *et al.*, 1998). By storing these relationships, a model can be fit rapidly despite a relatively large search space.

AAMs' focus on deformable models through training data makes this approach too computationally expensive. AAMs are more suited to distinction among different aircraft than actual detection.

III. DETECTION AND ESTIMATION METHODS

A. INTRODUCTION

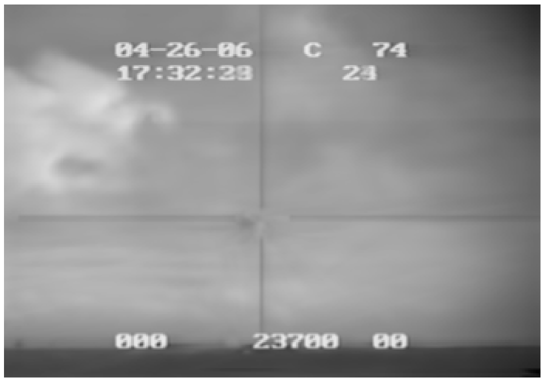
A general-purpose computer-vision algorithm does not exist for detecting aircraft in video. The primary purpose for this research is developing a detection method for aircraft in platform camera video. A combination of several methods allows estimation of aircraft position within a frame of video. Following is a summary of developed methods.

Initial development was performed on approaches flown in March 2006 aboard USS JOHN C. STENNIS (CVN-74).

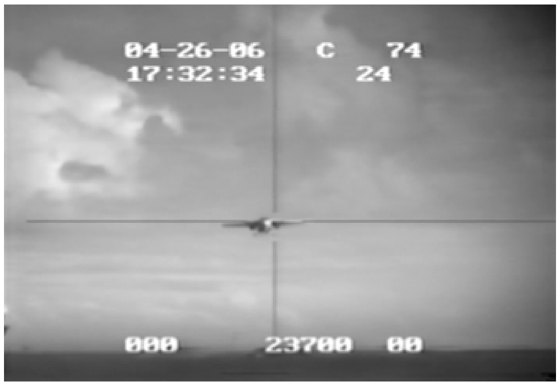
B. BACKGROUND DIFFERENCING

1. Background Model

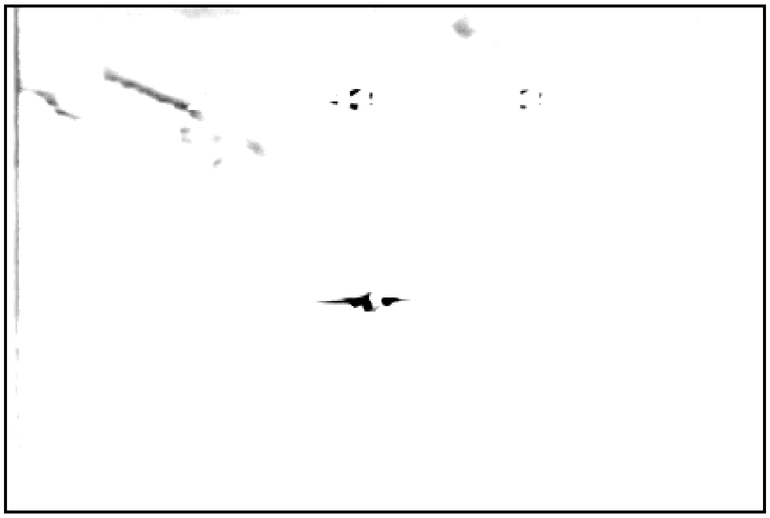
After an approach's completion, a learned background model emerges by averaging intensity values at every pixel location over all available frames. Subtracting individual frames from the average image breaks out motion in that frame.



(a) Average image for an S-3 approach



(b) Frame 512 from same approach



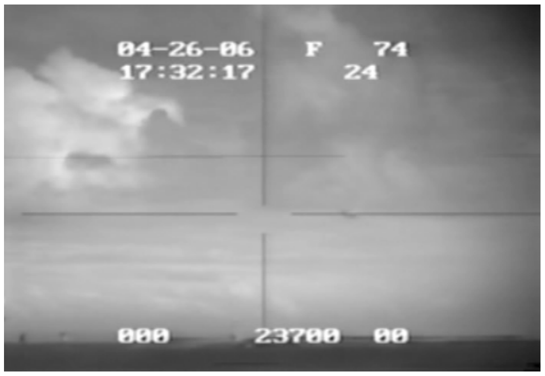
(c) Frame 512 subtracted from probability map for entire approach (inverse, contrast enhanced)

Figure 6. Example of probability map/average background differencing

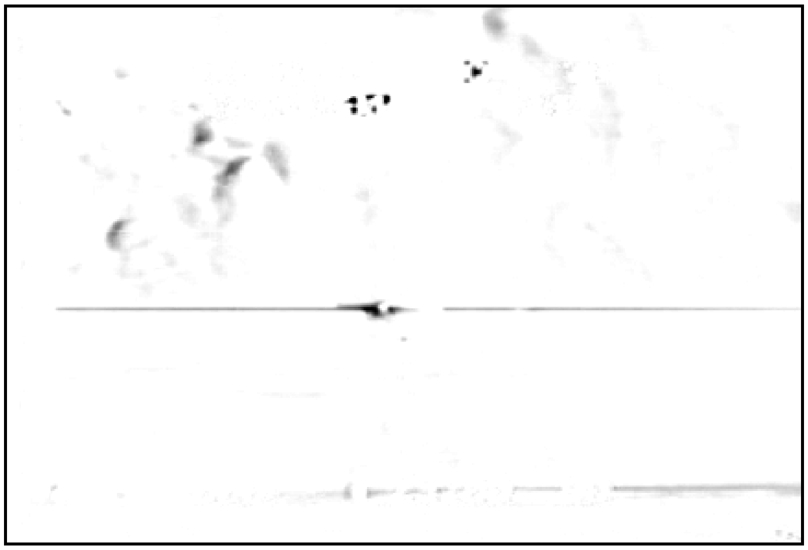
Figure 6c shows an aircraft clearly visible, with minor motion in the top third of the frame due to clouds or embedded text. The only disadvantage of a learned background image approach is the average image must be calculated after capturing all frames; therefore, this approach does not support real-time detection of aircraft in platform camera video.

2. First Frame

When searching for an aircraft in a series of video frames, it seems natural that a pixel-by-pixel difference between the first frame and a later frame would provide enough detail to identify aircraft motion during an approach.



(a) Frame 1 from an S-3 approach (b) Frame 398 from the same approach



(c) Frame 398 subtracted from frame 1 (inverse, contrast enhanced):

Darker response indicates greater difference

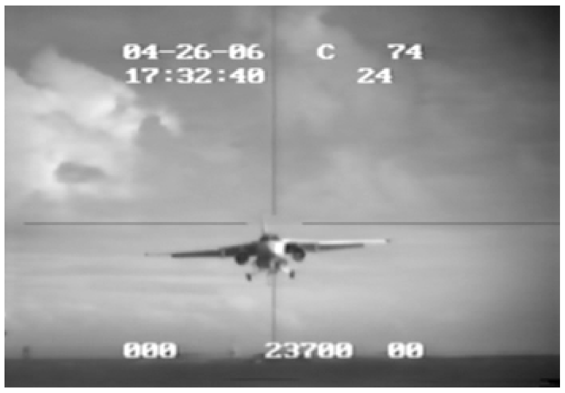
Figure 7. Example of "first-frame background differencing"

Figure 6c shows the movement of clouds during approach and certain changes in text appearing in the original frames. Although differencing to the first frame clearly indicates the presence of an aircraft to the human eye, there is too much information present for the computer to discern the aircraft.

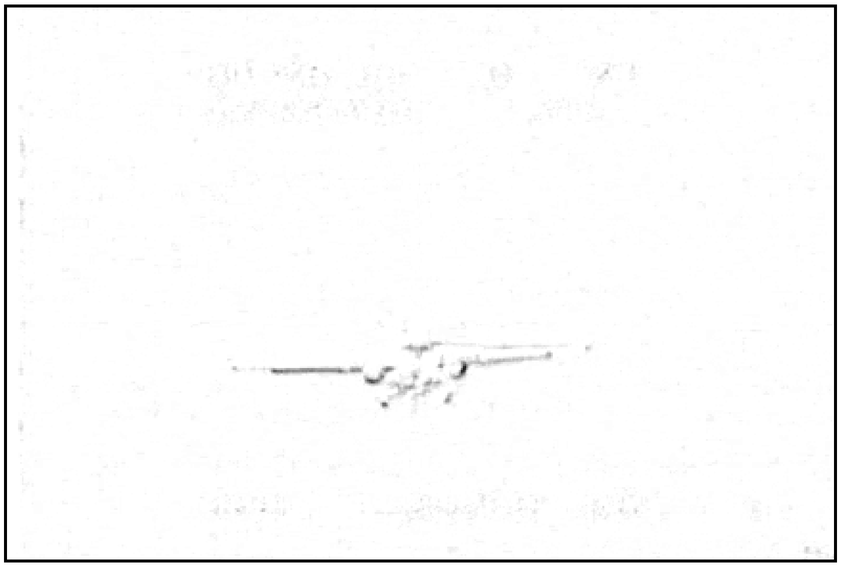
3. Frame-to-frame Differencing

Since "first frame differencing" appears to identify too much motion when comparing frames, it makes sense to compare a single frame to the one directly preceding it. Intuitively, there should be less motion in this comparison.

(a) Frame 676 of an S-3 approach



(b) Frame 677 of an S-3 approach



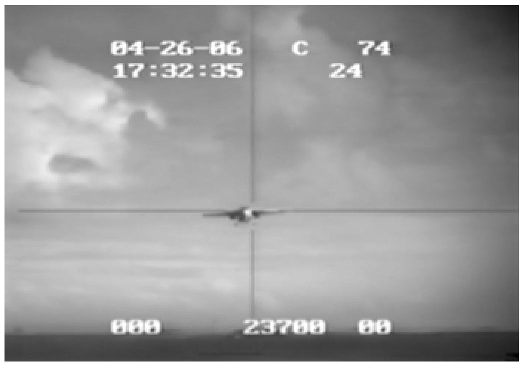
(c) Frame 677 subtracted from frame 676 (inverse, contrast enhanced)

Figure 8. Example of frame-to-frame background differencing

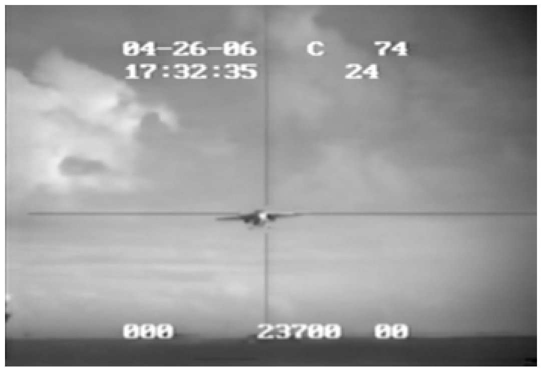
Figure 7c shows barely visible fragments of the aircraft's movement from the previous frame. Although an aircraft may move enough to detect motion in the previous frame, there is insufficient information to definitively locate the aircraft within the frame.

4. Nth-Previous Frame

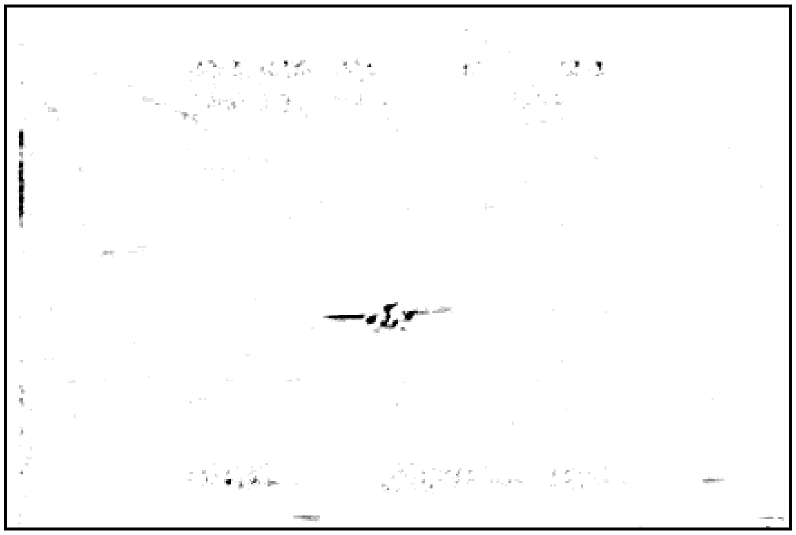
To refine the motion breakout between frames, the next logical step is to analyze a larger timestep than provided by the previous frame. "Nth-previous frame background differencing" focuses attention on the aircraft and desensitizes the effect of moving clouds or other transients in the video. For the platform camera, a seventh-previous frame subtraction provides a reasonable amount of aircraft motion, while concurrently limiting irrelevant motion in the frame.



(a) Frame 526 from an S-3 approach



(b) Frame 533 from same approach



(c) Frame 533 subtracted from frame 526 (inverse, contrast enhanced)

Figure 9. Example of "Nth-frame background differencing"

Figure 9c contains valuable location data of the aircraft. With the help of thresholding to create a binary image, this information is amplified and used to locate the aircraft within the frame.

The MATLAB® code used for "nth-frame background differencing" is provided in Appendix A.

C. THRESHOLDING

Thresholding creates a binary image from a grey-level or intensity image that reflects the greatest intensity change in a difference image.

For each pixel in the difference image, the intensity is compared to an appropriate threshold value. If the pixel's intensity exceeds the threshold, then the new value of that pixel is stored as binary one—255 in an 8-bit image, or 1.0 in a 32-bit floating point image. If the pixel's intensity does not exceed the threshold, however, the new value of that pixel is set to zero in both 8-bit and 32-bit images.

Since a single fixed threshold produced accurate results on all video sequences, an automatic grey-level threshold selection method such as that presented in Otsu (1979) is not needed. Thresholds changed less than one percent when porting our algorithm from MATLAB® to OpenCV, regardless of video analyzed.

When applied to the difference image of Figure 9c, thresholding amplifies motion information:

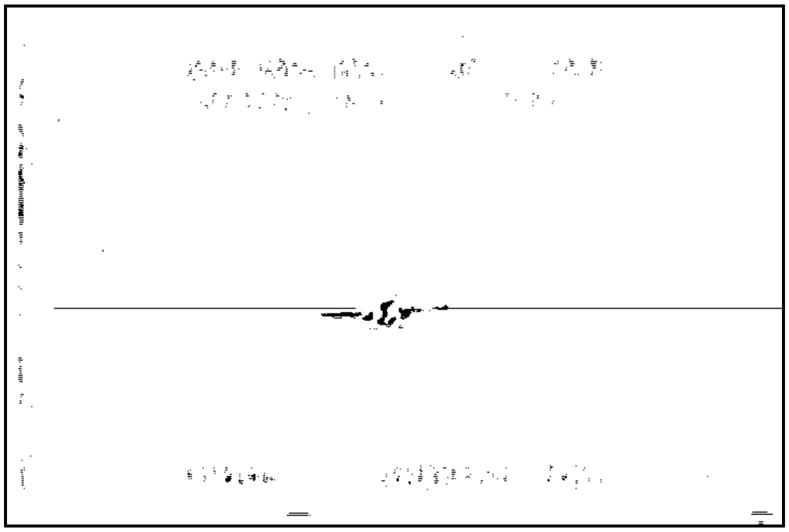


Figure 10. 7th-previous frame difference image (see Figure 9c) as binary with a threshold of 0.029 (inverse)

D. MORPHOLOGY

Simple morphology cleans up unwanted noise and motion within the frame. First, a closing is performed with a 20x20 disk-shaped structuring element to organize responses into larger blocks:

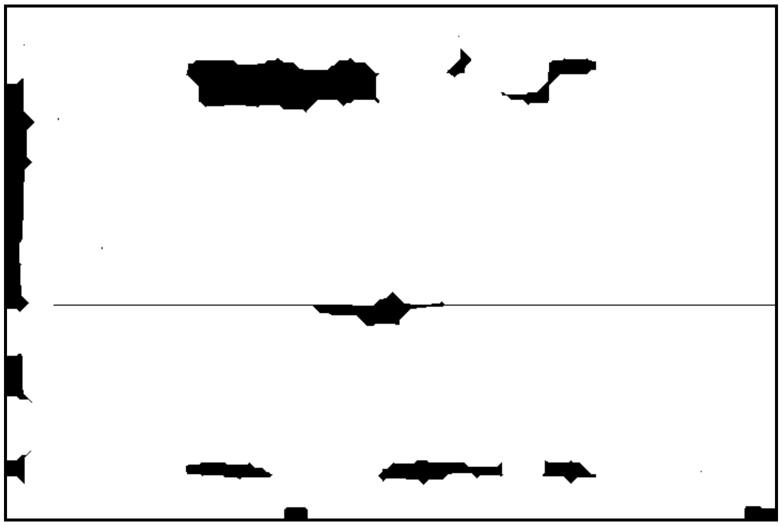


Figure 11. Image from Figure 10 after closing with a 20x20 disk-shaped structuring element

To eliminate unwanted lines and remaining noise, an opening is performed with a 5x5 disk-shaped structuring element:

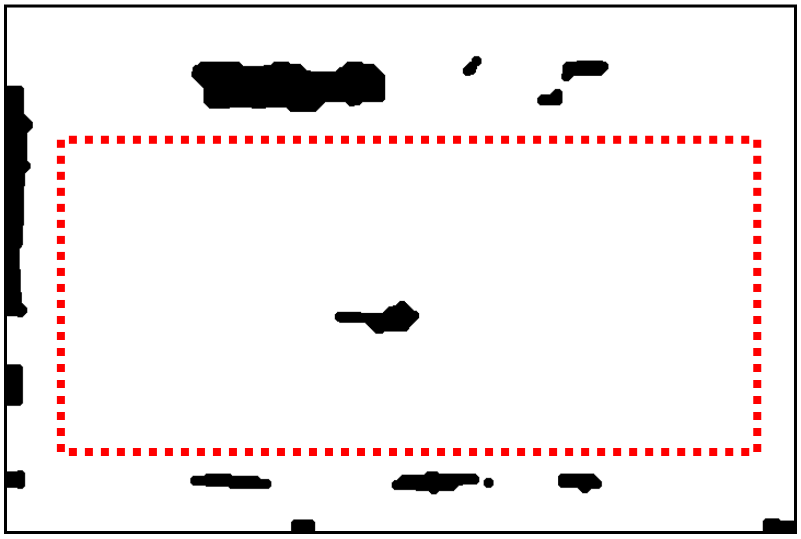


Figure 12. Image from Figure 11 after opening with a 5x5 disk-shaped structuring element; dotted line represents region of interest

E. DETERMINING AIRCRAFT LOCATION

The platform camera video contains known areas of undesired change—text along the top and bottom of the frame, as well as noise along the left side of the frame. By setting a region of interest (ROI), these changes are not interpreted as aircraft motion:

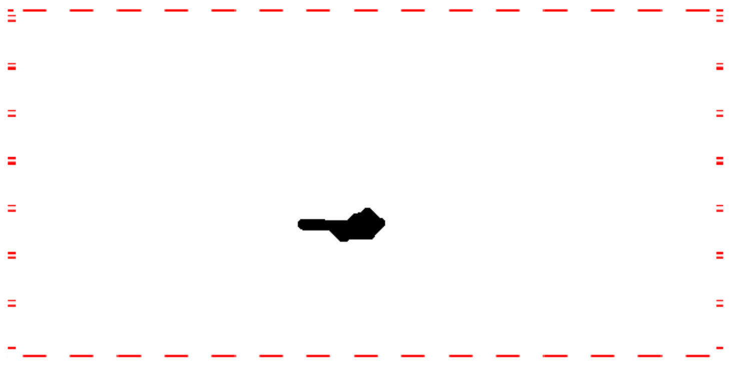


Figure 13. Region of interest applied to Figure 12

Aircraft location is calculated as the location of the largest connected region's center of mass within the ROI. Every location is stored from frame to frame, forming a history of detected aircraft locations. Each detected location may or may not represent the actual aircraft. To determine which detected location represents the aircraft's location, a method to eliminate outliers is implemented.

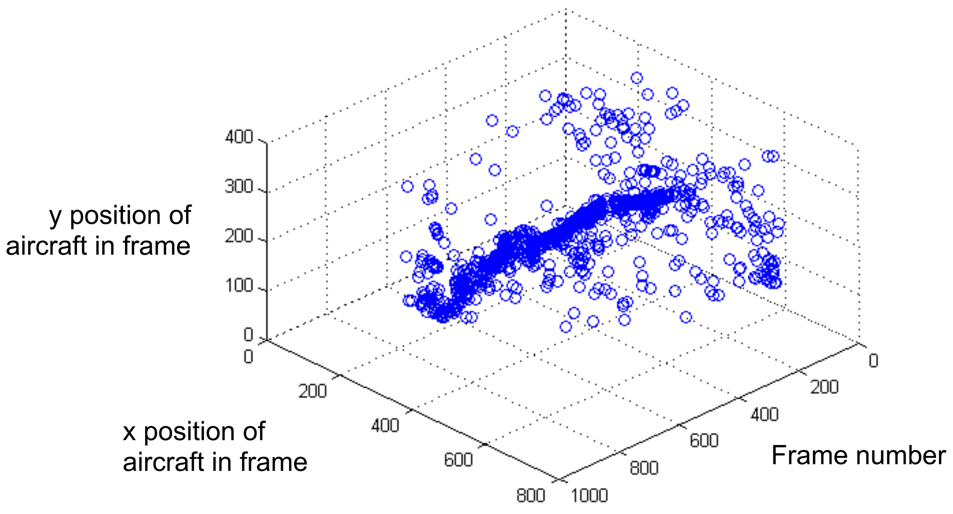


Figure 14. Raw data captured from an S-3 approach

F. ROBUST TEMPORAL FILTERING

To address the issue of outliers in the aircraft's predicted location, the Random Sample Consensus method is applied to the frame-based detections to provide temporal filtering. RANSAC is an iterative algorithm that fits a specified model to randomly sampled points. In the case of platform camera video, the model is a line through two or more points in two or more consecutive video frames, that corresponds to a locally-linear approximation of the aircraft's flight path. By comparing the remaining points to the specified model through multiple iterations, RANSAC determines which points fit the model, and which are outliers.

```
Determine:
    n – the smallest number of points required
    k – the number of iterations required
    t – the threshold used to identify a point that fits well
    d – the number of nearby points required to assert a line fits well
Until k iterations have occurred
    Draw a sample of n points from the data at random
    Fit specified model to that set of n points (See Appendix B, C)
    For each data point outside the sample
        Test the distance from the point to the line against t; if the
            distance from the point to the line is less than t, the
            point is close
    end
    If there are d or more points close to the line then there is a good
        fit. Refit the line using all these points.
end
Use the best fit from this collection, using the fitting error as a criterion.
```

Figure 15. General RANSAC algorithm (from Forsyth and Ponce, 2003).

By applying a Euclidean distance threshold of fifteen and analyzing 75 points at one time, the aircraft's path is represented easily in approach detection data. MATLAB® code for this implementation, including distance, fitting, and degenerate functions, is found in Appendices A2-A4.

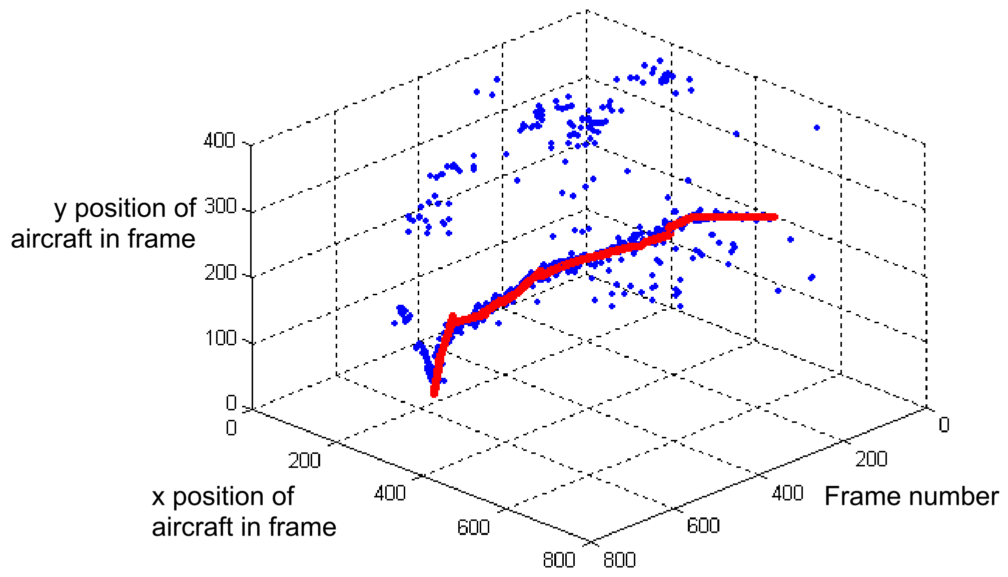


Figure 16. RANSAC algorithm applied to S-3 detection data; detected location data points shown in red

RANSAC, however, does not always provide a perfect solution. When large amounts of noise, or very spatially consistent noise such as other significant areas of contrast (i.e. dark clouds) affect approach data, the algorithm fits the data to a path other than the aircraft's.

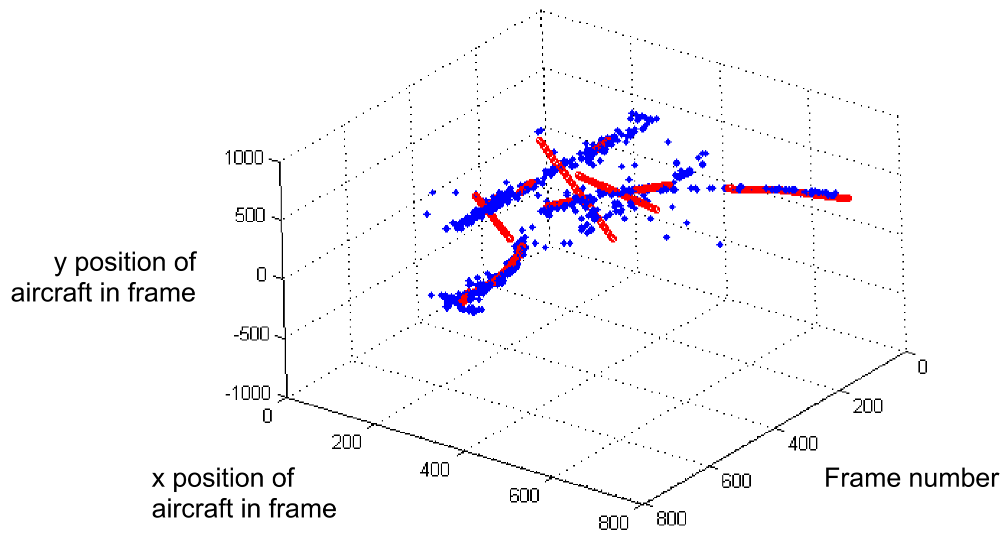


Figure 17. RANSAC algorithm applied to EA-6B detection data with large areas of contrast in background

In the nine approaches studied to develop our detection algorithm, aircraft were correctly detected in 61% of all frames. Other frames exhibited more prominent noise than aircraft features. Despite this high level of noise and outliers, temporal filtering with RANSAC is extracts the true motion of the aircraft in nearly every approach. This fact alone proves the power of a robust estimator function. Close to 100% accurate flight path detection is expected in combination with appearance-based aircraft detection (recommended future work).

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

A. DATA DESCRIPTION

The set of videos used to develop our algorithm consisted of nine approaches, courtesy of USS JOHN C. STENNIS (CVN-74). The approaches occurred during daylight hours on April 26th, 2006. Some approaches had cluttered backgrounds, mainly dark clouds; others showed more uniform backgrounds.

Platform camera video was transferred from Super Video Home System (most platform camera video is recorded in S-VHS) to Video Home System (VHS) aboard CVN-74. The VHS was then converted to a digital format using a WinTV-HVR-250 by Hauppage!.

Each approach was stored in Motion Picture Experts Group 2 format (MPEG-2): 720 pixels wide by 480 pixels high, interlaced at 29.97 frames per second (fps). Finally, frames were extracted to Windows Bitmap format (BMP) for analysis. No efforts were made to remove interlacing within the video.

B. IMPLEMENTATION

Results presented in this section were generated with the Image Processing Toolbox provided by MATLAB® (Copyright 1984-2005, The MathWorks, Inc.).

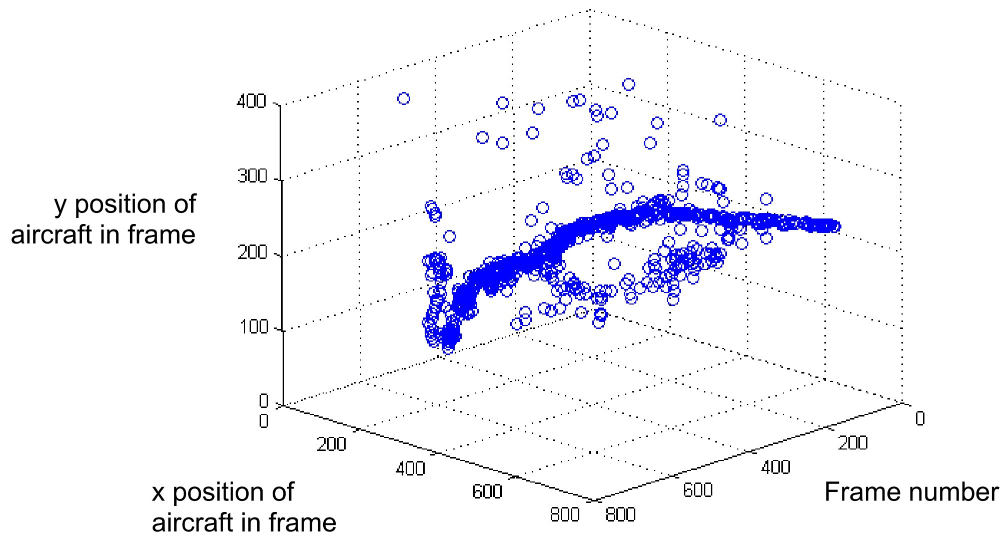
The author ported the algorithm to OpenCV for use in a real-time application. It is worth noting that MATLAB® and OpenCV toolboxes process images differently. For example, while methods discussed in Chapter III utilized structuring elements up to 20x20 within MATLAB®, the same structuring elements in OpenCV were as small as 3x3 to obtain similar results.

OpenCV also required fewer morphological operations than MATLAB®. Instead of multiple morphological operations like those presented in Chapter III, OpenCV often required only a simple erosion to detect aircraft motion in “nth-frame background difference” images.

OpenCV code is provided in Appendix B for this implementation.

C. NORMAL APPROACH

Algorithms presented in Chapter III produced realistic representations of aircraft approaches during normal daytime operations. Detected motion regions per frame appear as blue circles or blue dots. RANSAC-filtered aircraft locations are shown in red:



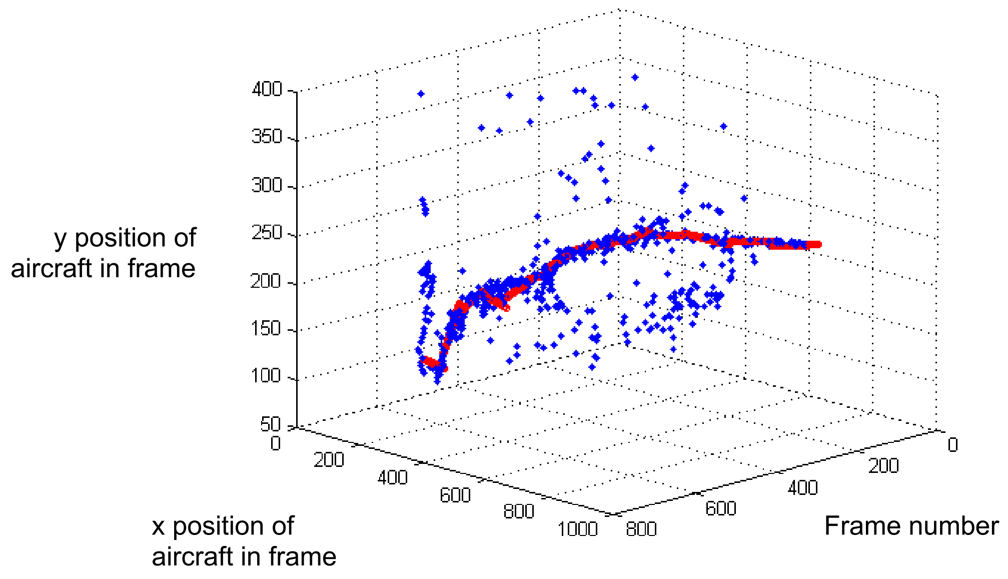
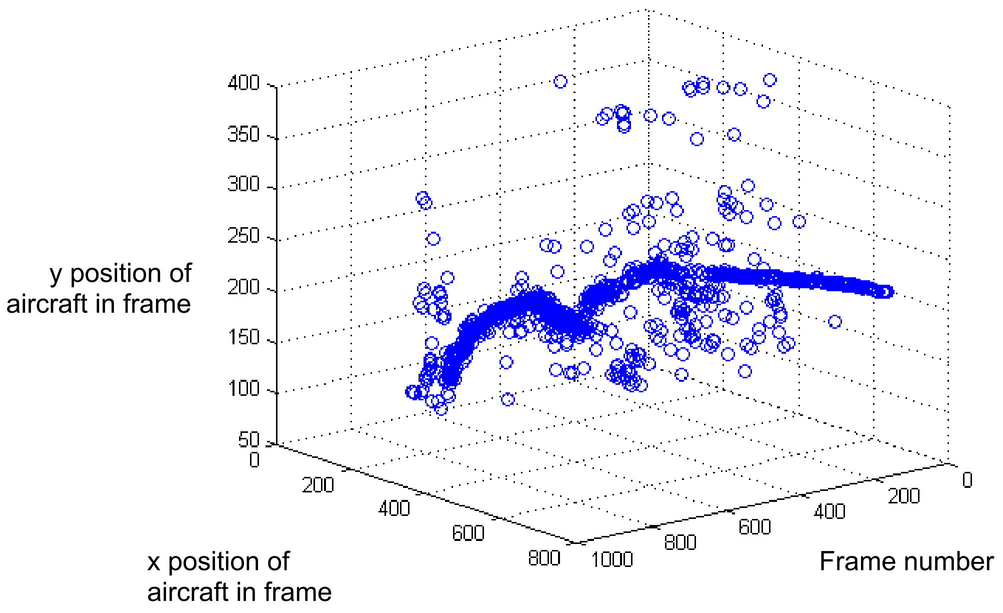


Figure 18. EA-6B Frame-based detection data (top) and post-temporal analysis (bottom); detected motion regions per frame in blue; RANSAC-filtered aircraft locations in red



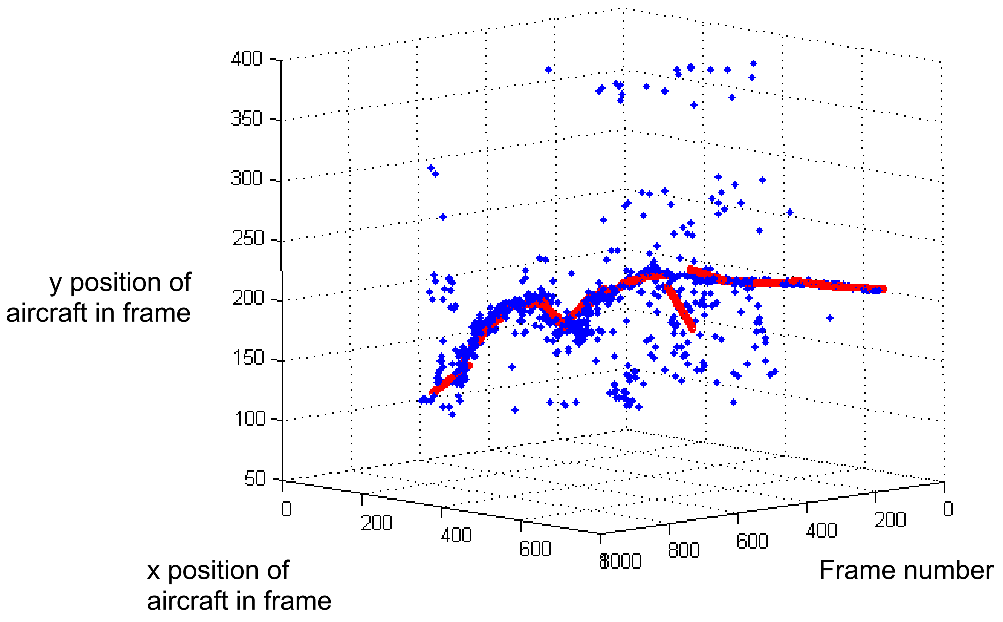
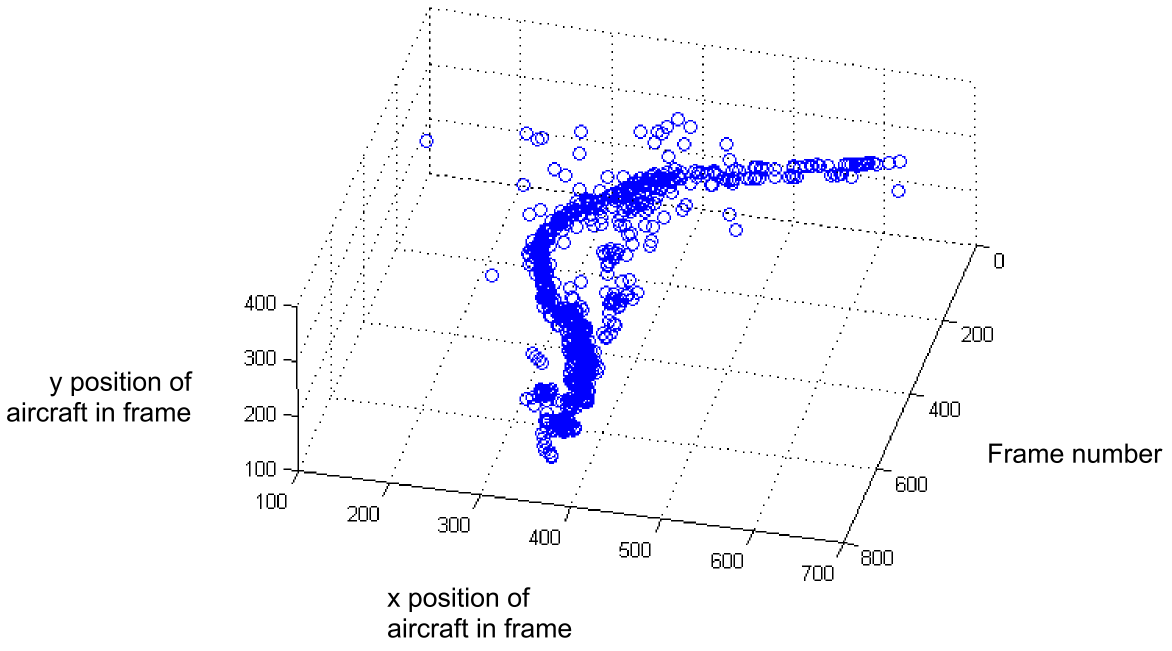


Figure 19. S-3 Frame-based detection data (top) and post-temporal analysis (bottom); detected motion regions per frame in blue; RANSAC-filtered aircraft locations in red

D. OVERSHOOTING APPROACH

Our detection algorithm produced accurate approach representations during an aircraft overshoot—landing area centerline is at an x-position of 360:



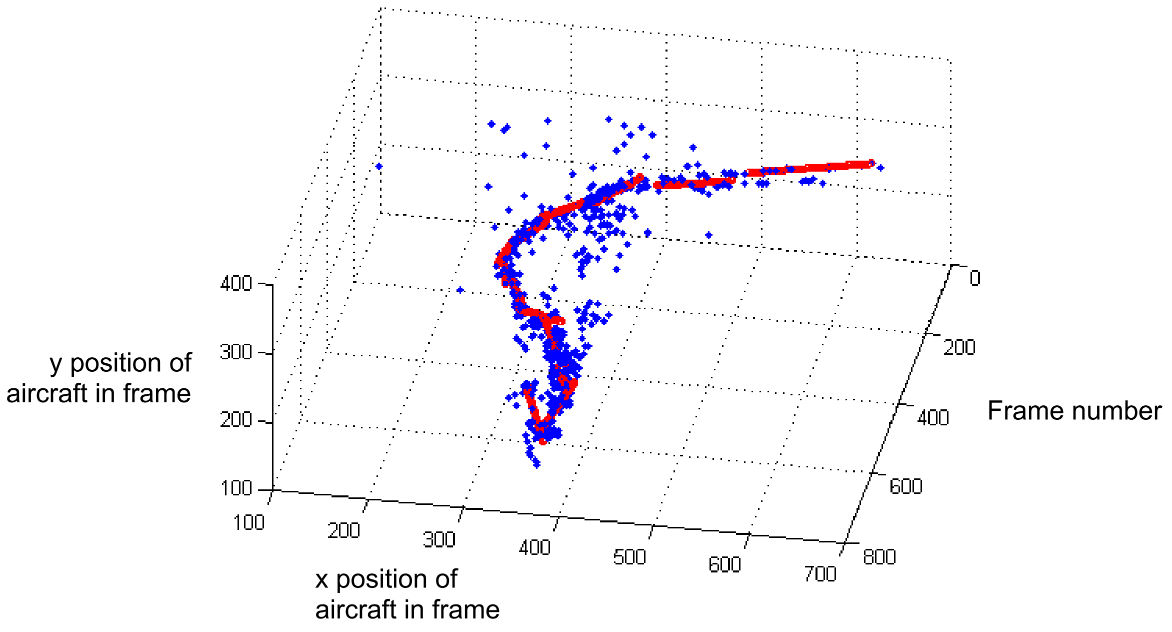
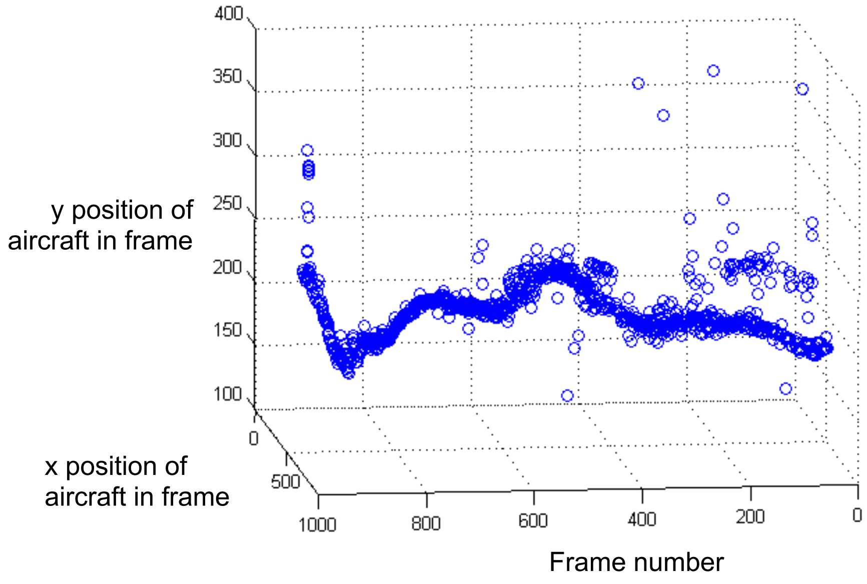


Figure 20. EA-6B Frame-based detection data (top) and post-temporal analysis (bottom); detected motion regions per frame in blue; RANSAC-filtered aircraft locations in red

E. WAVEOFF

Our detection algorithm accurately represented a C-2 waved off in close during an approach. Notice the abrupt increase in altitude toward the end of the approach (toward left side of figures below, following frame 850):



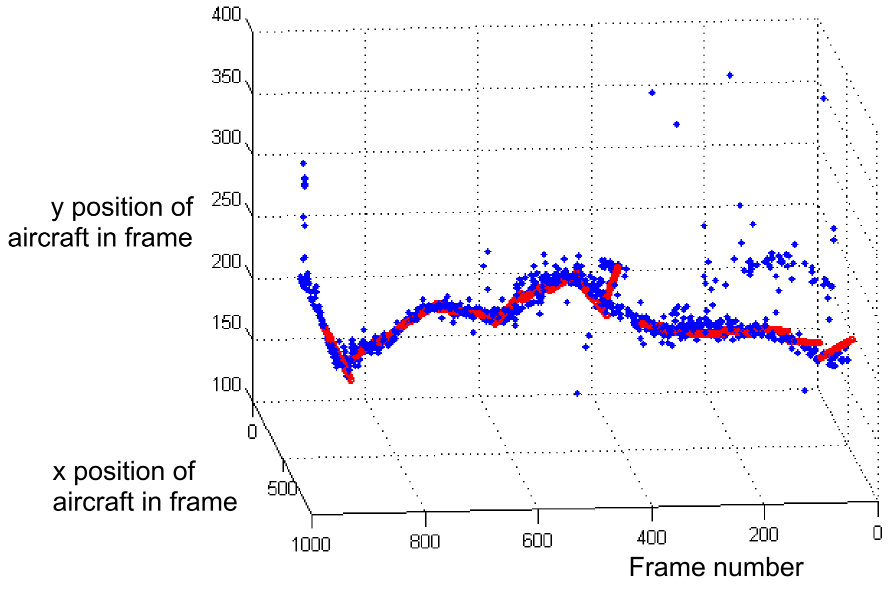


Figure 21. C-2 Frame-based detection data (top) and post-temporal Analysis (bottom); detected motion regions per frame in blue; RANSAC-filtered aircraft locations in red; frame number (and time) increase from right to left

Additional approach analyses can be found in Appendix C.

F. USABILITY EVALUATION

Naval aviators, having seen only approach analyses presented in Chapter IV, easily recognize characteristics present in post-temporal analyses. An informal survey of two LSOs and two pilots verified appropriate trends. Figure 19 clearly shows a settle on start, Figure 20 clearly shows an overshoot, and Figure 21 clearly shows an LSO-commanded waveoff in close.

Although study participants identified approach characteristics and trends within a two-dimensional images, three-dimensional imaging provides additional insights into approaches, and allows for more accurate re-creations of respective approaches.

G. FIELD LANDING EXTENSION

Our airfield implementation consists of an Allied Vision Technologies (AVT) Guppy F-046 Firewire camera, with attached manual telephoto 10X zoom lens (11.0mm – 110mm), situated slightly above runway elevation at a safe distance from Monterey Peninsula Airport (MRY). Captured frames are ported to OpenCV and analyzed.

Since our capture frame rate is reduced to 15 fps, the background difference image is changed to third-previous frame to represent the same amount of motion present in 29.97 fps seventh-previous frame difference images. Figure 22 depicts a screen capture of an aircraft taking off from MRY in March 2007. The green rectangle represents an appropriate ROI; the small red box represents the algorithm’s detected location of the aircraft.

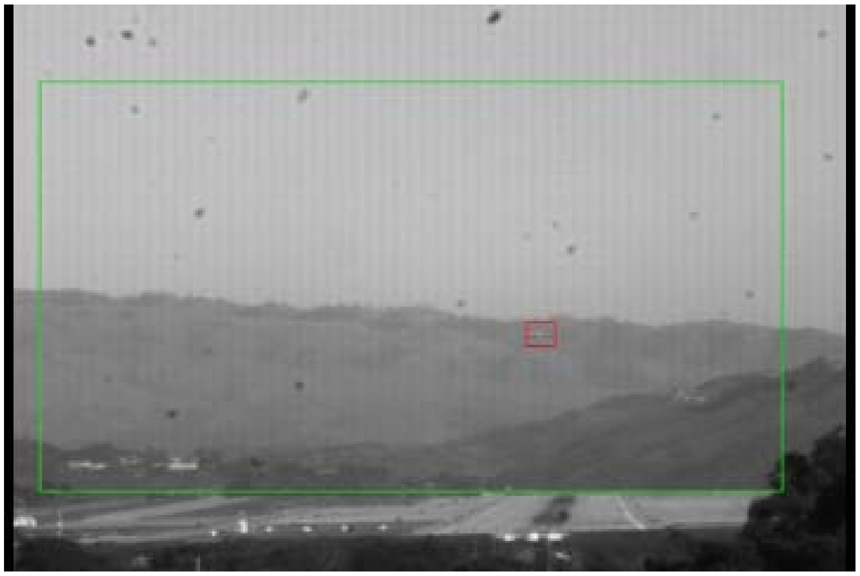


Figure 22. Aircraft taking off from MRY, March 2007; aircraft is successfully detected and marked with a small red box

Frame-based detection data appears in Figure 23; a bird flying across the frame is visible toward top of graph.

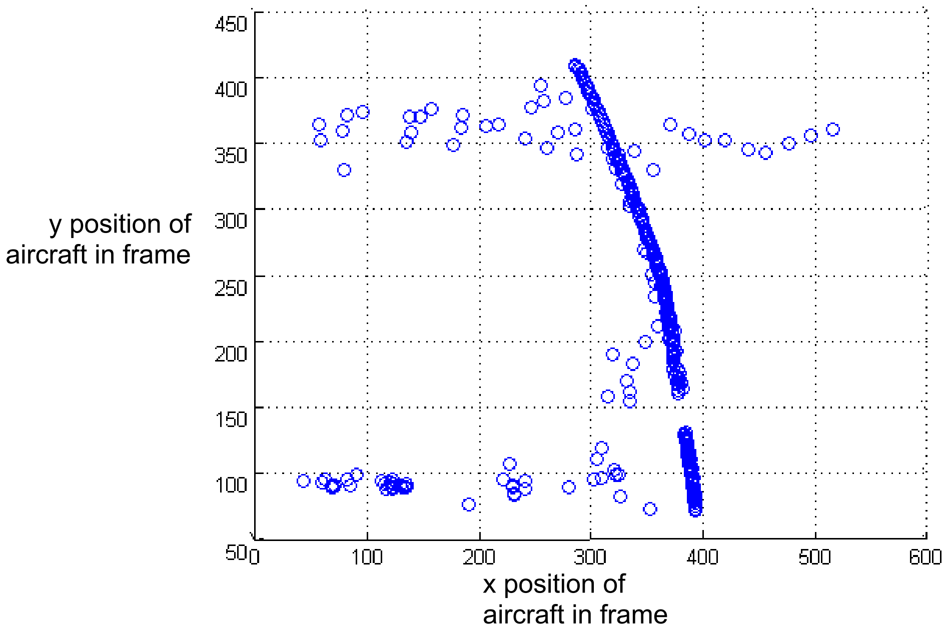


Figure 23. Frame-based detection data from MRY takeoff

When applying RANSAC to this data with a Euclidian-distance threshold of ten and analyzing 75 points at a time, the bird's path is eliminated, clearly showing the aircraft's path (Figure 24).

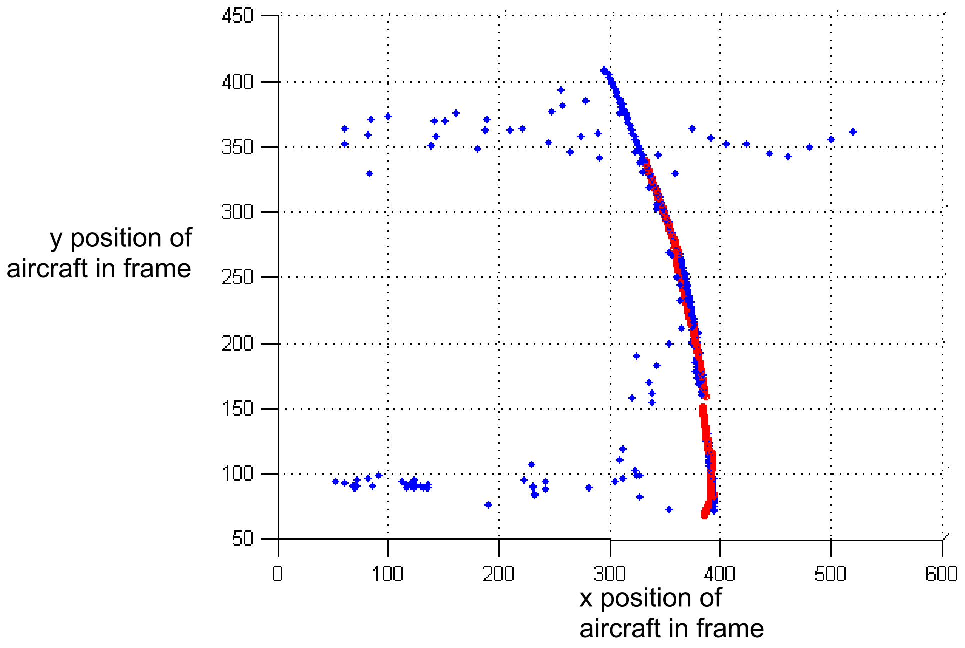


Figure 24. Post-temporal analysis of MRY takeoff

A sample video of this takeoff is available at the website address listed in Chapter V.

H. ROBUST ESTIMATOR FUNCTION

The author defines detection rate as the number of flight path inliers divided by the total number of data points collected for a single approach. The robust estimator function identifies 61% or more inliers in two thirds of the nine approaches. The highest detection rate in any single approach was 81%; the lowest detection rate was 27% on an excessively noisy video with extremely poor quality.

Regardless of detection rate, the quality of temporal filtering provided by RANSAC cannot be overstated. With as little as a 53% detection rate in video, the estimator function is able to extract a more than 90% accurate approach representation. When detection rate is higher (as in Figures 23 and 24, representing 74% detection rate), noise is eliminated entirely. A bird that flew across the frame and extraneous motion close to the ground are correctly ignored, thanks to the robustness of RANSAC.

THIS PAGE INTENTIONALLY LEFT BLANK

V. APPROACH RE-CREATION AND VISUALIZATION

A. EXTENSIBLE MARKUP LANGUAGE (XML)

Recommended by the international standards organization for the World Wide Web, XML is a general-purpose markup language that supports numerous applications.

XML derives strength from its ability to share data across different systems, and is used primarily on internet platforms. Data stored in XML format is easily read by both humans and machines.

Due to its strength in data storage, XML is an ideal language to capture and store approach data for future review. Its extensibility also provides potential for easy translation into trend analyses for individual pilots covering any time period in the aviator's career.

A sample XML structure used to store raw approach data is provided in Appendix D.

B. EXTENSIBLE 3D (X3D)

Extensible 3D (X3D) is the International Organization for Standardization's (ISO) standard for real-time three-dimensional computer graphics. X3D replaced the Virtual Reality Modeling Language (VRML), and allows creation of virtual worlds that can be viewed by anyone with access to the World Wide Web.

The ability to view 3D virtual worlds from any web browser makes X3D the perfect data format to visually re-create approaches based on captured flight path detection data stored in XML format. This re-creation allows the pilot to "re-fly" any recorded approach whenever he chooses.

C. CONVERTING DETECTION DATA INTO X3D

Detection data is captured as (x,y) coordinates within a frame of video. Therefore, a simple matrix transformation is needed to convert captured data into the world coordinates conventionally used in X3D's virtual world. Detection data is then stored in XML format, each point containing aircraft altitude, displacement from centerline, and distance from ship.

An Extensible Stylesheet Language Transformation (XSLT) sets up the virtual world for an approach visualization. First, a carrier-centered coordinate system is created. Next, a generic aircraft carrier is drawn. Finally, each data point is attached to an animated camera that acts as the pilot's eye for "re-flying" the approach.

A proof of concept application was developed in the context of this thesis to demonstrate the feasibility and visual appearance of the after-action review visualization. Figure 25 shows the pilot's view during a re-created landing approach. Figure 26 shows a re-creation of the Improved Fresnel Lens Optical Landing System (IFLOLS), which indicates the aircraft's position in space relative to the ideal flight path.

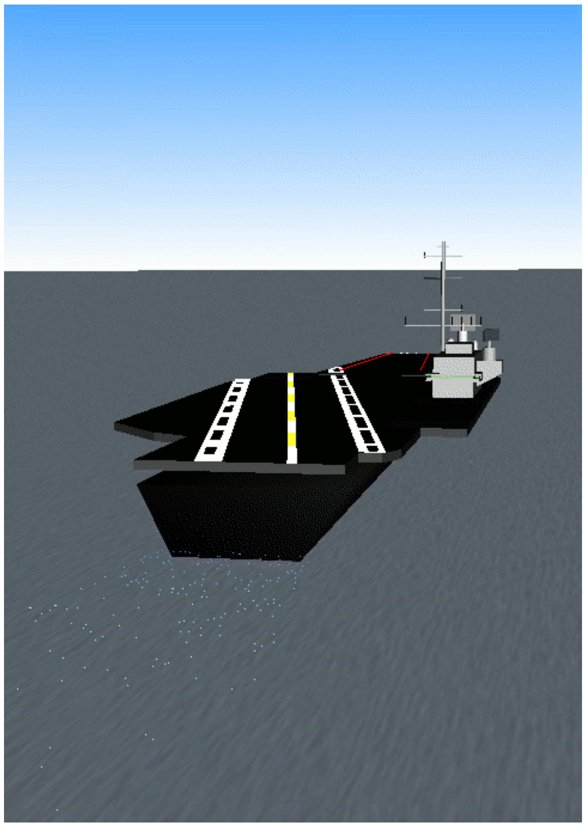


Figure 25. Sample screenshot of X3D approach re-creation. "pilot's eye" view

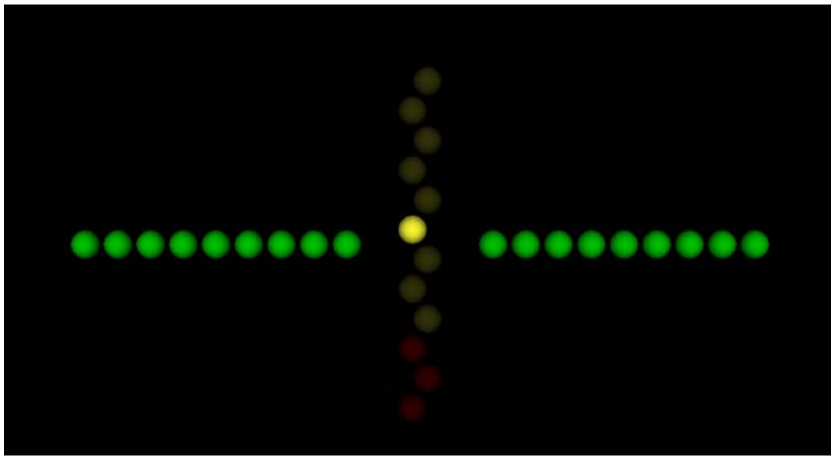


Figure 26. X3D representation of IFLOLS

The XSLT file used to transform approach data for this proof of concept is listed in Appendix D.

D. IMPACT ON PILOT PERFORMANCE

Normally, a Naval aviator must rely on his memory and verbal comments from an LSO to re-create an approach. Additional objective feedback combined with a recognizable visual representation of an approach is expected to greatly enhance pilot performance. A comprehensive study is recommended for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

This research provides a computer vision-based analysis of platform camera video to both pilots and landing signal officers. Approach situations were explored, resulting in valuable information stored for further review to aid in training, approach analyses, and pilot self-improvement. Robust temporal filtering proved critical in extracting appropriate data from video. The following paragraphs address the research questions from Chapter I individually.

1. Aircraft Detection and Tracking

Our algorithm successfully detected and tracked aircraft in platform camera video with two degrees of freedom through several dynamic approach scenarios. A third degree of freedom was artificially created through video timesteps; however, insufficient information was captured to calculate actual range information of any given aircraft from the ship.

2. Usability of Generated Data

Collected data for this research produced usable representations of aircraft approaches. Naval aviators presented with temporally unfiltered data were able to describe key characteristics of the respective approach. This confirms the intuitiveness of 3D-approach representations. A short glance was all that was needed to recognize regular, overshooting, and waved-off approaches. Subjects were able to report additional detail about presented approaches.

3. Static Airfield Plausibility

Tests confirmed suitability of the algorithm to tracking from an airfield. Approach information was as consistent as that presented in Chapter IV. A standardized camera location needs to be established, however, to provide LSOs with consistent data during FCLPs, since our airfield test cases focused on civilian airfields.

B. FUTURE WORK

Our research demonstrated impressive potential to improve training for both pilots and Naval aviators throughout the Fleet. Several recommended areas for expanding this research follow:

1. Develop Robust Cascades of Simple Features

Currently, our methods do not exploit appearance-based detection of the aircraft but instead, only look for motion within video. One method showing promise recently for object detection is Viola and Jones's idea (2001) of boosted cascades of simple features. For the purposes of an detecting aircraft, one might consider a more robust implementation of multiple cascades—one each for different types of aircraft, and several for different orientations. This has potential to improve detection accuracy and speed within platform camera video, and could provide more efficient aircraft detections, which would, in turn, lead to more precise approach representations.

2. Standardize Static Airfield Implementation

Although data collected produced acceptable approach representations, a more thorough exploration of airfield implementation should occur. The best location for the camera must be determined and feasibility verified at multiple locations. The overall priority: camera should remain portable and capture accurate data.

3. Develop a Nighttime Tracking Algorithm

This research was limited to Case I (daytime) operations. Nighttime approaches could provide equally valued feedback. Detecting and tracking an aircraft after sunset, however, creates unique challenges not addressed in this thesis. Infrared video can solve many limitations of visible-light sensor cameras.

4. Impact on an Aviator's Training and Performance

A comprehensive study to determine the effects of additional objective feedback on an aviator's performance and whether, in the aviator's mind, this extra information adds value, is a reasonable sequel to this research. A similar study could confirm parallel effects regarding LSO training.

C. ADDITIONAL RESOURCES

Sample videos and research figures are available online at http://www.movesinstitute.org/. Please forward any questions or comments about this research to the author at this email address: ryan.yusko@gmail.com.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: SELECTED MATLAB CODE

1. NTH FRAME DIFFERENCE

Following is our “nth frame difference” image implementation, used to process individual difference images to locate an aircraft:

```
%processI010(x,y) - version 0.10
%x is char prefix of image files you are analyzing
%y is frame to start analysis
%returns an array of [y a b] where a,b correspond to x,y in
%       "normal-thinking" coordinates (LL origin)
%   ex. [y a b] from a 720x480 frame would be considering
%               the LL corner of the frame to be the origin
%       where (a,b) corresponds to the coordinates of the
%       center of a centroid found in the image
%       after morphology.  the frame (y) is repeated to
%       prevent confusion in later processing
function ary = processI010(x,y)
%x is prefix for filename in current directory
%y is frame number (up to 999) to process

ary = [];

%global/setup variables, i.e. ROI
y1box = 100;
y2box = 400;
x1box = 50;
x2box = 670;

%form3digit returns a text representation of passed number,
%as a 3 digit integer (i.e. '5' would be returned as '005')
imNum = form3digit(y);

%read an image, convert it to grayscale, and
%determine its size
z = imread([x imNum '.bmp']);
z = rgb2gray(z);
[rows cols] = size(z);

%nth frame differencing; in this case, we are using
%n = 7
y2 = y-7;

%form the integer value for the second image
imNum2 = form3digit(y2);

%read second image, and convert it to grayscale
z2 = imread([x imNum2 '.bmp']);
z2 = rgb2gray(z2);

%calculate difference image
```

```
diff = z - z2;

%set up morphology kernels
se20 = strel('disk',20);
se5 = strel('disk',5);
se2 = strel('disk',2);

%create binary thresholded image 'd'
%in an 8-bit image, .02 is equivalent to ~5.12
d = im2bw(diff,.02);

%perform morphology with 20x20 disk within ROI
cl = imclose(d(y1box:y2box, x1box:x2box),se20);

%perform morphology with 5x5 disk
%ROI already integrated into "closed" image
op = imopen(cl,se5);

%calculate regionprops, and set up index for center of mass
props = regionprops(bwlabel(op), 'basic');
[num one] = size(props);
idx = -1;

if (num == 0) %if no regions found, perform "smaller" morphology
    cl = imclose(d(y1box:y2box, x1box:x2box),se5);
    op = imopen(cl,se2);
    props = regionprops(bwlabel(op),'basic');
    [num one] = size(props);
end

%if we found regions via "large" or "small" morphology,
%return center of mass of largest area found
if (num ~= 0)
    for ix = 1:num
        a = props(ix).Centroid(1)+x1box;
        b = rows - (props(ix).Centroid(2)+y1box);
        ary = [ary ; y a b];
    end
end

%show final image
imshow(op);
```

2. RANSAC ANALYSIS OF DATA

detectFP analyzes datasets, and applies RANSAC to determine which points are outliers, and which are the aircraft:

```
%function detectFP
%
%Analyzes data passed using RANSAC algorithm.
%Data is analyzed [seglen] points at a time, and stepped
% through with a step value of [seglen_use]
%
%@param x            - data to be analyzed
%@param thresh       - threshold used by RANSAC to keep or throw
%                      out fitted model
%@param seglen       - number of points to be analyzed at a time
%@param seglen_use   - value used to step through data
function [out inliers] = detectFP(x, thresh, seglen, seglen_use)
%x - array of xyz predictions gained from doPredict

%set default for seglen_use if only 3 args passed
if (nargin == 3)
    seglen_use = seglen;
end

%determine size of data
[rows cols] = size(x);

ix = 0;

%for ix = 0:loop
while (ix*seglen_use+seglen <= rows)
    %get a viable solution from ransac
    %MATLAB RANSAC implementation courtesy of:
    %
    % Peter Kovesi
    % School of Computer Science & Software Engineering
    % The University of Western Australia
    % pk at csse uwa edu au
    % http://www.csse.uwa.edu.au/~pk
    %
    % May      2003 - Original version
    % February 2004 - Tidied up.
    % August   2005 - Specification of distfn changed to allow model
    %                 fitter to return multiple models from which the
    %                 best must be selected.
    [M,in]=ransac(x(ix*seglen_use+1:ix*seglen_use+seglen,:),
        'fitF','distF','degenF',2,thresh,0);

    start = max([1 floor((seglen-seglen_use)/2)]);
    finish = start+seglen_use-1;

    %generate inliers structure
    inliers(1,ix+1) = length(in);
    %generate output data
    for jx = start:finish
        xIn = ix*seglen_use+jx; %x(in(1),1);
        t = (xIn - M(1,1))/M(1,2);
        out(ix*seglen_use+jx,1) = xIn; % ix*10+jx;
        out(ix*seglen_use+jx,2) = M(2,1)+t*M(2,2);
        out(ix*seglen_use+jx,3) = M(3,1)+t*M(3,2);
    end

    %increment ix for next swing through loop
    ix = ix + 1;
end

%adjust output to match data collected
out = out(start:size(out),:,:);
```

3. RANSAC FITTING FUNCTION

fitF is used by the RANSAC algorithm to fit a model in an appropriate format:

```
% fitF returns a model in the form
%
%       [ p_x   d_x ]
%       [ p_y   d_y ]
%       [ p_z   d_z ]
%
% This model describes a line between two data
% points x1 and x2.

function M = fitF( x )

M = zeros(3,2);
x1 = x(1,:);
x2 = x(2,:);

%store point for our model
M(1,1)=x1(1); %p_x from x1
M(2,1)=x1(2); %p_y from x1
M(3,1)=x1(3); %p_z from x1

%generate direction vector for our model
M(1,2)=x2(1)-x1(1); %d_x from x2-x1
M(2,2)=x2(2)-x1(2); %d_y from x2-x1
M(3,2)=x2(3)-x1(3); %d_z from x2-x1

end
```

4. RANSAC DISTANCE FUNCTION

distF is used by RANSAC to determine which data points are inliers as specified by a threshold:

```
%distF
%
%Returns inliers that fall within the distance
%specified by t
function [inliers,Mout] = distF(M,x,t)

Mout = M;
[rows,cols] = size(x);
hold=1;

%iterate through data
for ix = 1:rows
    d = distFromM(M,x(ix,:));

    %if distance is less than threshold 't',
    %store it as an inlier
    if (d <= t)
        inliers(hold)=ix;
        hold=hold+1;
    end
end
```

5. RANSAC DEGENERATE FUNCTION

degenF is used by RANSAC to determine whether random points are valid, or whether the appropriate amount have been selected:

```
% degenF returns a boolean value based on criteria
% for fitting a line to our model.  Returns false if
% size is not equal to 2.  Returns false if dataset
% contains two identical points
function r = degenF(x)

%assume function returns false until corrected
r = false;

%determine size of data passed
[rows,cols] = size(x);

%if we got 2 points, store them for evaluation
if (rows == 2)
    x1 = x(1,:);
    x2 = x(2,:);
end

%check for validity
if (rows ~= 2)
    r = true;
elseif (x1 == x2)
    r = true; %dataset contains two identical points
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: SELECTED OPENCV CODE

1. REAL-TIME DETECTION ALGORITHM

detectOne() is the workhorse function for our real-time detection application. Please note simplification of morphological operations as compared to the MATLAB® implementation presented in Appendix A.

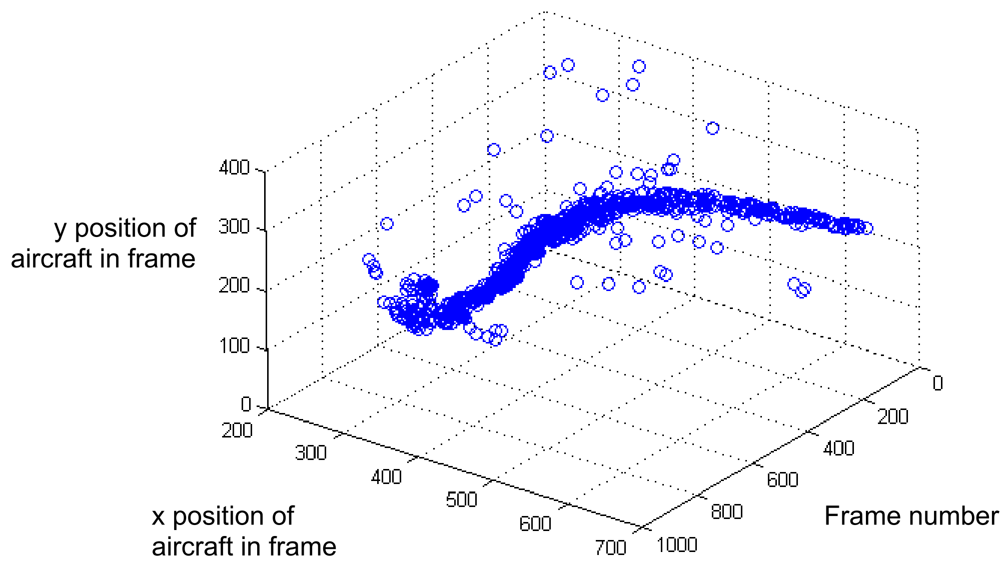
```
//*******************************************************************
// Function:          detectOne ( IplImage*, IplImage*, const int )
// Return Value:      int
// Parameters:        im1 - first image to analyze (earlier frame)
//                    im2 - second image to analyze (later frame)
//                    delay - delay value passed to cvWaitDelay
// Purpose:           Attempts to detect aircraft from two images
//                    provided.
//                    Returns negative int if escape key is pressed.
//*******************************************************************
int detectHelper::detectOne( IplImage* im1, IplImage* im2,
                             const int delay )
{
    static int processed = 0;
    static int thresh = 7;
    IplImage* diff = cvCreateImage( cvSize(im1->width,im1->height),
        IPL_DEPTH_8U, 1 );
    IplImage* color = cvCreateImage( cvSize(im1->width,im1->height),
        IPL_DEPTH_8U, 3 );

    //create raw difference image
    cvSub( im2, im1, diff );

    //create binary image with defined threshold difference
    //set threshold in third argument
    //threshold is 7 unless changed by key input (see switch statement
    //   below)
    cvThreshold( diff, diff, thresh, 255., CV_THRESH_BINARY );

    //perform morphology
    cvErode( diff, diff, se2 );
    //cvErode(diff,diff,se3);
    //cvDilate(diff,diff,se2);

    //prepare images for display; if showing binary image, convert the
    //difference image; if not, just convert the regular frame
    //images have to be RGB to display reticle and ROI "nicely"
    if ( showBinary )
    {
        cvCvtColor( diff, color, CV_GRAY2RGB );
    }
    else
    {
        cvCvtColor( im1,color, CV_GRAY2RGB );
```

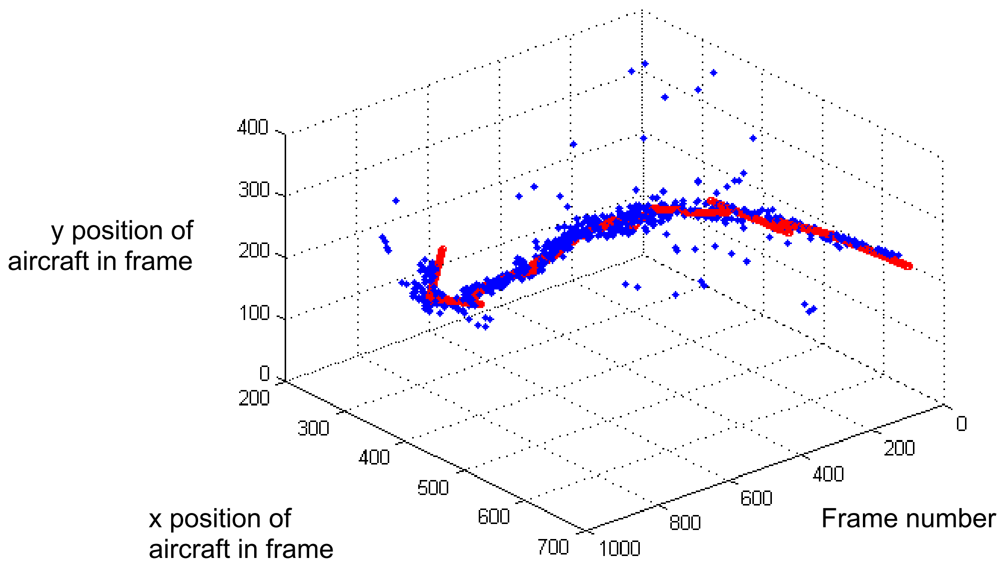
```
}


//set image ROI to eliminate unwanted noise
cvSetImageROI( diff, cvRect( point.x, point.y, ROI_WIDTH,
       ROI_HEIGHT ));

//calculate moments
cvMoments( diff, moments, 1 );

//calculate center of mass
double x_c = moments->m10 / moments->m00;
double y_c = moments->m01 / moments->m00;

int buff = 10; //this is +- pixels for determining detection
               //reticle

if ( showDetect )
{
    cvRectangle( color,
        cvPoint((int)x_c-buff+point.x,(int)y_c-buff+point.y),
        cvPoint((int)x_c+buff+point.x,(int)y_c+buff+point.y),
        CV_RGB(255,0,0) );
}

if ( showROI )
{
    cvRectangle( color, point,
        cvPoint(point.x+ROI_WIDTH,point.y+ROI_HEIGHT),
        CV_RGB(0,255,0));
}

//show the final image
cvShowImage("detect",color);

int result = 0, key = cvWaitKey(delay);

//perform cleanup & memory management
cvReleaseImage(&im1);
cvReleaseImage(&im2);
cvReleaseImage(&diff);
cvReleaseImage(&color);

//process keypress, if any
switch ( key )
{
case 'r':
case 'R':
    showROI ^= 1;
    break;
case 'd':
case 'D':
    showDetect ^= 1;
    break;
case 'i':
case 'I':
        showBinary ^= 1;
        break;
    case '1':
        thresh = 1;
        break;
    case '2':
        thresh = 2;
        break;
    case '3':
        thresh = 3;
        break;
    case '4':
        thresh = 4;
        break;
    case '5':
        thresh = 5;
        break;
    case '6':
        thresh = 6;
        break;
    case '7':
        thresh = 7;
        break;
    case '8':
        thresh = 8;
        break;
    case '9':
        thresh = 9;
        break;
    case '0':
        thresh = 10;
        break;
    case 27: //ESC key pressed
        result = -1;
        break;
    default:
        break;
    }//switch(key)

    return result;
}//detectOne()
```

THIS PAGE INTENTIONALLY LEFT BLANK

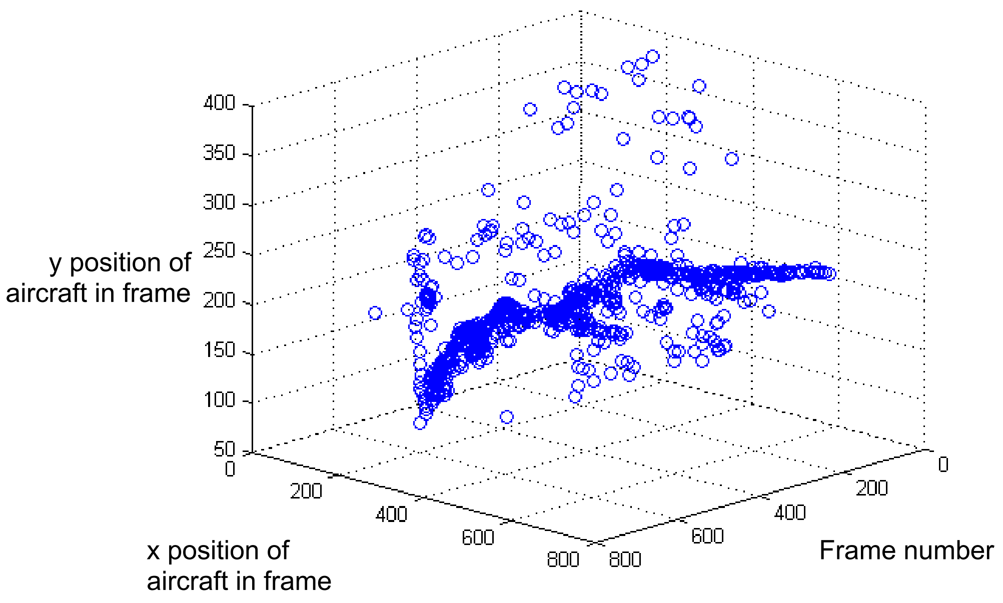
APPENDIX C: ADDITIONAL APPROACH ANALYSES

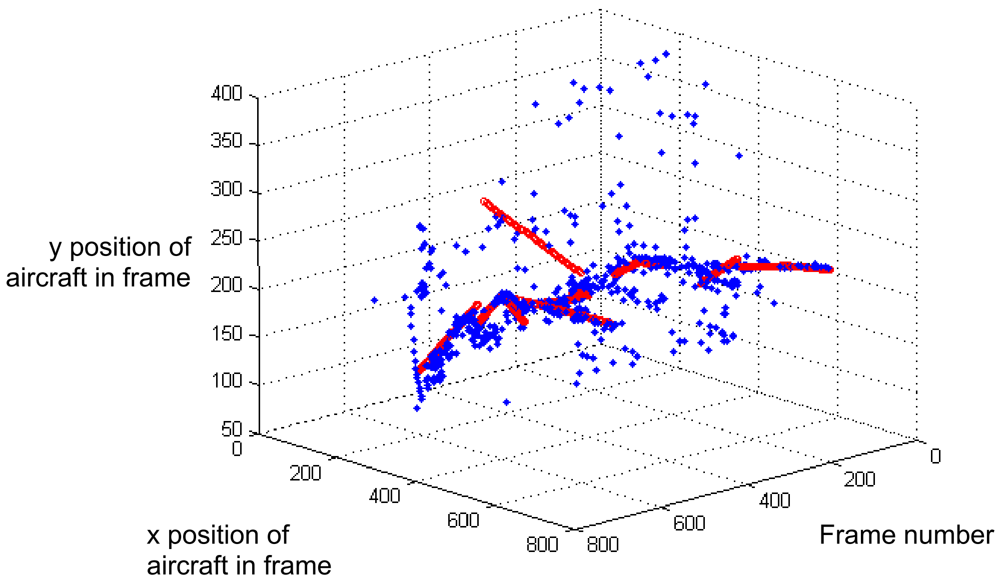
C-2 (1705) frame-based detection data and post-temporal analysis:



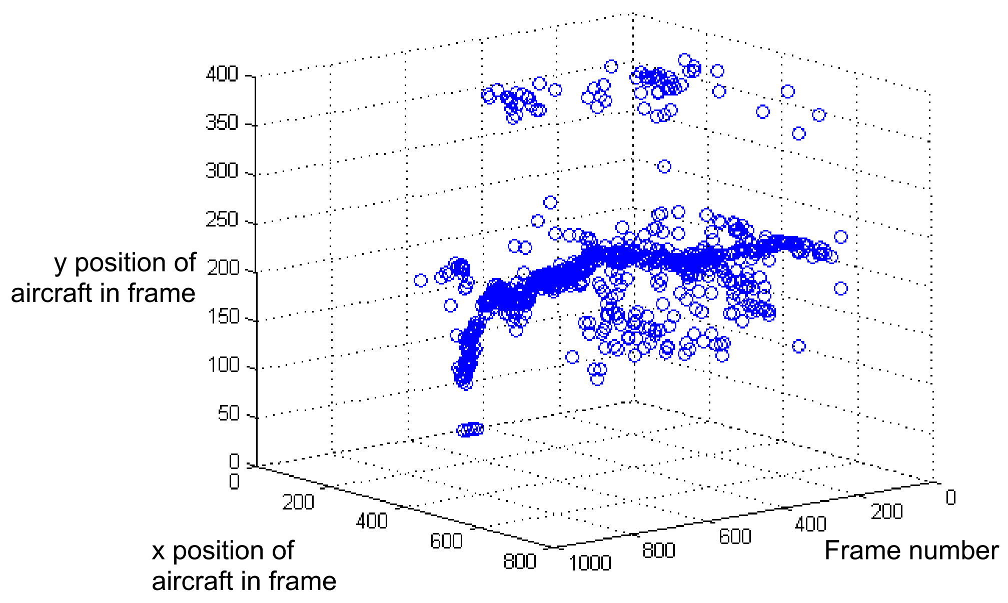


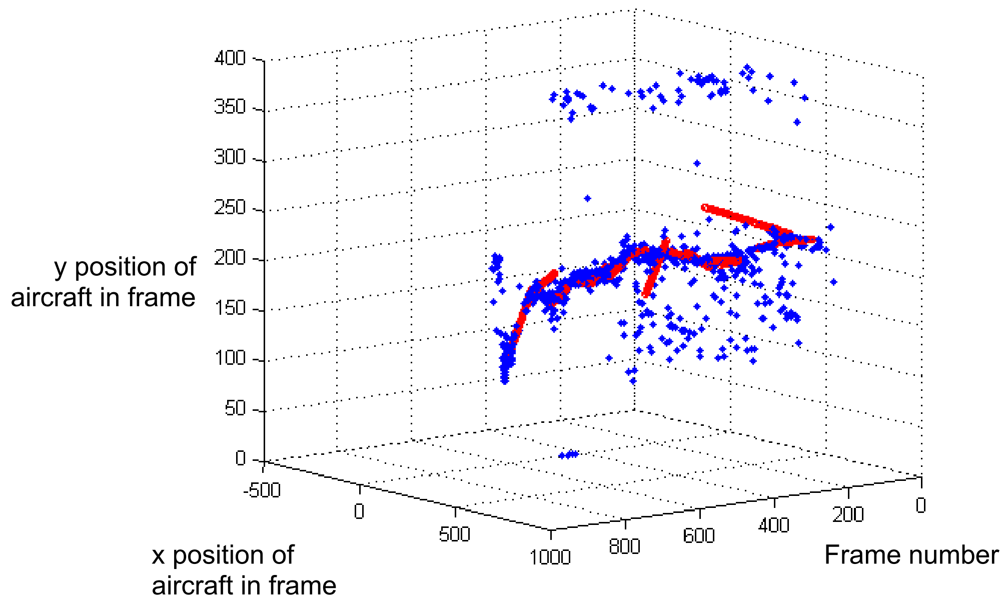
F/A-18 (1710) frame-based detection data and post-temporal analysis:



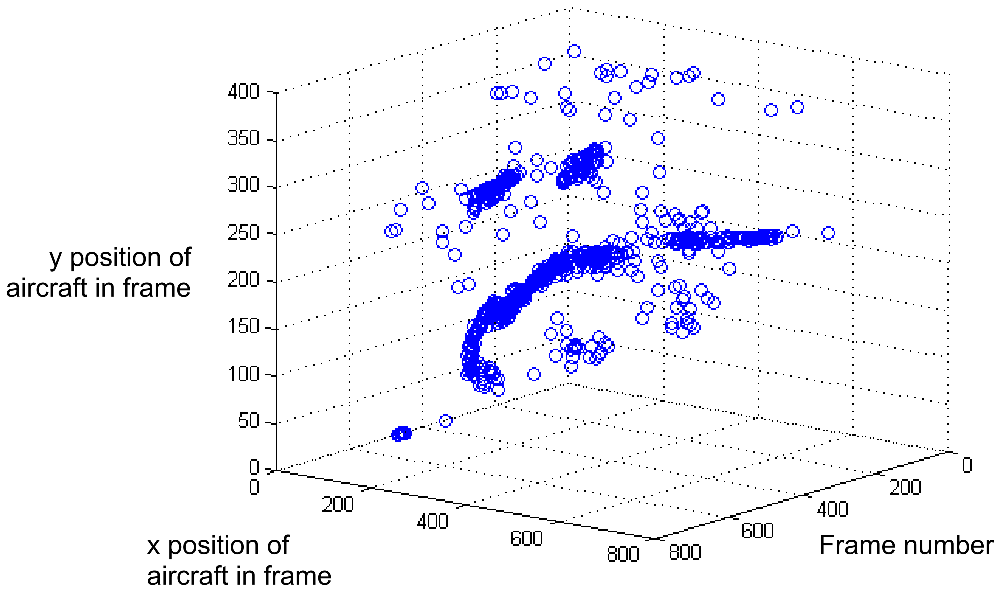


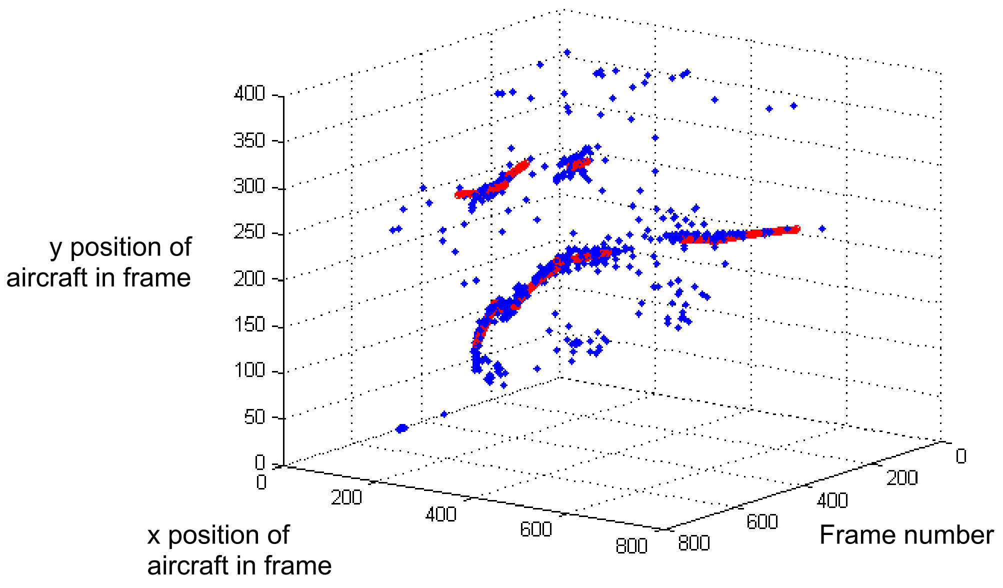
F/A-18 (1727) frame-based detection data and post-temporal analysis:





F/A-18 (1734) frame-based detection data and post-temporal analysis:





APPENDIX D: APPROACH RE-CREATION FILE SAMPLES

1. SAMPLE XML FORMAT FOR FLIGHT PATH STORAGE

The following XML representation of approach data was used in a proof of concept for approach re-creation. Aircraft location is specified in world coordinates: altitude in feet (mean sea level), displacement in feet from centerline (negative is left of centerline, while positive is right of centerline), and distance from touchdown point in feet. The rotation value specifies “head turn,” and is used to keep the “pilot’s eye” looking at the ship when left or right of centerline.

```
<approach ver="0.2">
  <date time="1641">20070301</date>
  <approachData groove="8">
    <data>
      <frame number="1" rotation="-0.207747"/>
      <key value="0"/>
      <time value="1172796093"/>
      <location altitude="151.725" disp="-207.747" dist="1304.28"/>
    </data>
    <data>
      <frame number="2" rotation="-0.206876"/>
      <key value="0.00423729"/>
      <time value="1172796093"/>
      <location altitude="151.089" disp="-206.876" dist="1298.82"/>
    </data>
    <data>
      <frame number="3" rotation="-0.206006"/>
      <key value="0.00847458"/>
      <time value="1172796093"/>
      <location altitude="150.45" disp="-206.0" dist="1293.35"/>
    </data>
 (…)
    <data>
      <frame number="236" rotation="-7.64392e-005"/>
      <key value="0.995763"/>
      <time value="1172796101"/>
      <location altitude="1.40" disp="-0.0764" dist="20.5398"/>
    </data>
    <data>
      <frame number="237" rotation="-5.61096e-005"/>
      <key value="1"/>
      <time value="1172796101"/>
      <location altitude="1.03" disp="-0.056" dist="15.07"/>
    </data>
  </approachData>
</approach>
```

2. SAMPLE XSLT - APPROACH DATA TO X3D

The following XSLT creates an interactive virtual world of an aircraft's approach to the aircraft carrier. A generalized aircraft carrier is used; approach data from an XML file in the format provided in Appendix D1 is translated to an animated camera view representing the "pilot's eye."

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "http://www.web3d.org/specifications/x3d-3.1.dtd"
                    "./dtd/x3d-3.1.dtd">
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"
omit-xml-declaration="no"/>
<xsl:template match="approach">

<!--Warning:  transitional DOCTYPE in source .x3d file -->
<X3D profile="Immersive" version="3.1"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance"

xsd:noNamespaceSchemaLocation="http://www.web3d.org/specifications/x3d-
3.1.xsd">
  <head>

  <Scene>
  <Transform rotation=".56 .56 .61 2.04" translation="1000 0 100">
    <Viewpoint description="startup"/>
    </Transform>
    <Transform DEF="motion" rotation=".56 .56 .61 2.04" \
      translation="0 0 0">

    <xsl:element name="Transform">
    <xsl:attribute name="DEF">motionRotation</xsl:attribute>
    <Viewpoint description="Approach Motion"/>

    <xsl:element name="OrientationInterpolator">
      <xsl:attribute name="DEF">viewpointRotation</xsl:attribute>
      <xsl:attribute name="key">
        <xsl:for-each select="approachData/data/key">
          <xsl:value-of select="@value"/>
          <xsl:text> </xsl:text>
        </xsl:for-each>
      </xsl:attribute>
      <xsl:attribute name="keyValue">
        <xsl:for-each select="approachData/data/frame">
          <xsl:text>0 1 0 </xsl:text>
          <xsl:value-of select="@rotation"/>
          <xsl:text> </xsl:text>
        </xsl:for-each>
      </xsl:attribute>
    </xsl:element>
```

```
        <!-- <PositionInterpolator key="0 0.5 1" keyValue="3 4 5"/> -->
    <xsl:element name="PositionInterpolator">
      <xsl:attribute name="DEF">viewpointTranslation</xsl:attribute>
      <xsl:attribute name="key">
        <xsl:for-each select="approachData/data/key">
          <xsl:value-of select="@value "/>
            <xsl:text> </xsl:text>
        </xsl:for-each>
      </xsl:attribute>
      <xsl:attribute name="keyValue">
        <xsl:for-each select="approachData/data/location">
          <xsl:value-of select="@distance"/><xsl:text> </xsl:text>
          <xsl:value-of select="@displacement"/><xsl:text> </xsl:text>
          <xsl:value-of select="@altitude+15"/><xsl:text> </xsl:text>
          <!--here, we add hook to eye for more accurate display -->
        </xsl:for-each>
      </xsl:attribute>
    </xsl:element>
      <xsl:element name="TimeSensor">
      <xsl:attribute name="DEF">clock</xsl:attribute>
      <xsl:attribute name="loop">true</xsl:attribute>
      <xsl:attribute name="cycleInterval">
        <xsl:value-of select="approachData/@groove"/>
      </xsl:attribute>
      </xsl:element>
    <!--<TimeSensor DEF="clock" loop="true" cycleInterval="18.0"/> -->
    <ROUTE fromField="value_changed" fromNode="viewpointTranslation" \
      toField="set_translation" toNode="motion"/>
    <ROUTE fromField="value_changed" fromNode="viewpointRotation" \
      toField="set_rotation" toNode="motionRotation"/>
    </xsl:element>
    </Transform>
  <!--Original x:y:z 1:1:1 translation was -12:7.35:0. Use
      these numbers to apply new scales of aircraft carrier to "appear"
      realistic. -->
    <Transform rotation="0 0 1 2.943" scale="15 15 15" translation="- \
      180 110.25 0">
    <Inline url="&quot;Independence.wrl&quot;&#10;&quot;\
       http://www.xxx.com/x3d/Independence.wrl&quot;"/>
    </Transform>
    <!-- Uncomment the following transform to view coordinate axes :)
    <Transform scale="15 15 15">
    <Inline url="&quot;CoordinateAxesVrml.wrl&quot;"/>
    </Transform> -->
    <ROUTE fromField="fraction_changed" fromNode="clock" \
      toField="set_fraction" toNode="viewpointTranslation"/>
    <ROUTE fromField="fraction_changed" fromNode="clock" \
      toField="set_fraction" toNode="viewpointRotation"/>
  </Scene>
</X3D>

</xsl:template>
</xsl:stylesheet>
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Ballard, D. H. “Generalizing the Hough Transform to Detect Arbitrary Shapes.” Pattern Recognition 13.2 (1981). 111-22.

Cootes, T. J., G. J. Edwards and C. J. Taylor. "Active Appearance Models." European Conference on Computer Vision, 1998.

Fathy, M., and M. Y. Siyal. "An Image Detection Technique Based on Morphological Edge Detection and Background Differencing for Real-Time Traffic Analysis." Pattern Recognition Letters 16.12 (1995): 1321-30.

Forsyth, David, and Jean Ponce. Computer Vision : A Modern Approach. Upper Saddle River, N.J. ; London: Prentice Hall, 2003.

Gonzalez, Rafael C., and Richard E. Woods. Digital Image Processing. 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2002.

Intille, Stephen S., James W. Davis and Aaron F. Bobick. "Real-Time Closed-World Tracking." CVPR97. November, 1996.

Otsu, Nobuyuki. "A Threshold Selection Method from Gray-Level Histograms." IEEE Transactions on Systems, Man, and Cybernetics 9.1 (1979): 62-6.

Shapiro, Linda G., and George C. Stockman. Computer Vision. Upper Saddle River, NJ: Prentice Hall, 2001.

Viola, Paul, and Michael Jones. "Robust Real-Time Object Detection." Second International Workshop on Statistical and Computational Theories of Vision. Vancouver, Canada, 2001.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Space and Naval Warfare Systems Center
   San Diego, California

4. Naval Air Systems Command
   Patuxent River, Maryland

5. Naval Safety Center
   Norfolk, Virginia